

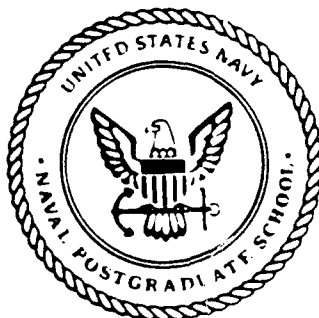
AD-A246 941



2

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
MAR 05 1992
S D D

THESIS

COMPARISON OF UNCONSTRAINED AND
CONSTRAINED CALIBRATION METHODS

by

Michael J. Wiegand

JUNE 1991

Thesis Advisor:

Morris R. Driels

Approved for public release: Distribution is unlimited

92-05298



92 3 02 050

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No 0704-0188	
1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: Distribution is unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b. OFFICE SYMBOL (If applicable) ME		7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State and ZIP Code) Monterey, CA 93943-5000			7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)			10. SOURCE OF FUNDING NUMBER		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
11. TITLE (Include Security Classification) COMPARISON OF UNCONSTRAINED AND CONSTRAINED CALIBRATION METHODS					
12. PERSONAL AUTHORS MICHAEL J. WIEGAND					
13a. TYPE OF REPORT Master's Thesis		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) JUNE 1991	
15. PAGE COUNT 81					
16. SUPPLEMENTARY NOTATION The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block numbers)		
FIELD	GROUP	SUB-GROUP	Robot calibration		
19. ABSTRACT (Continue on reverse if necessary and identify by block numbers) The idea of using a passive end point motion constraint to calibrate robot manipulators is of particular interest because no measurement equipment is required. The accuracy attained using this method is compared to the accuracy attained by an unconstrained calibration using computer simulated measurements. A kinematic model is established for each configuration using the Denavit-Hartenberg methodology. The kinematic equations are formulated and are used in the computer simulated calibration to determine the actual kinematic parameters of the manipulator. The results are discussed in terms of the effect of measurement noise and the number of experimental observations on the accuracy of parameter identification.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT XX UNCLASSIFIED/UNLIMITED SAME AS RPT DTIC USERS				21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Morris R. Driels			22b. TELEPHONE (Include Area Code) (408) 646-3383		22c. OFFICE SYMBOL ME/Dr

Approved for public release: Distribution is unlimited

Comparison of Unconstrained and
Constrained Calibration Methods

by

Michael J. Wiegand
Lieutenant, United States Navy
B.S., Ag. Eng., Michigan State University, 1983

Submitted in partial fulfillment of the
requirements for the degree of

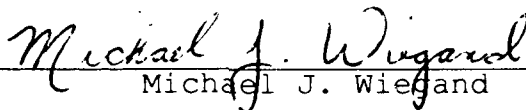
MASTER OF SCIENCE IN
MECHANICAL ENGINEERING

from the

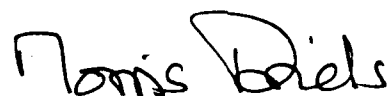
NAVAL POSTGRADUATE SCHOOL

JUNE 1991

Author:


Michael J. Wiegand

Approved by:



Morris R. Driels, Thesis Advisor



Anthony J. Healey, Chairman
Department of Mechanical Engineering

ABSTRACT

The idea of using a passive end point motion constraint to calibrate robot manipulators is of particular interest because no measurement equipment is required. The accuracy attained using this method is compared to the accuracy attained by an unconstrained calibration using computer simulated measurements. A kinematic model is established for each configuration using the Denavit-Hartenberg methodology. The kinematic equations are formulated and are used in the computer simulated calibration to determine the actual kinematic parameters of the manipulator. The results are discussed in terms of the effect of measurement noise and the number of experimental observations on the accuracy of parameter identification.



Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	THEORY	7
	A. THE DENAVIT-HARTENBERG METHOD WITH MODIFICATIONS	7
	B. PARAMETER IDENTIFICATIONS	12
III.	CALIBRATION METHODS	18
	A. UNCONSTRAINED METHOD	18
	B. CONSTRAINED (BALLBAR) METHOD	24
IV.	DISCUSSION	33
V.	CONCLUSIONS	45
	APPENDIX A: ZXSSQ	46
	APPENDIX B: COMPUTER PROGRAMS	48
	LIST OF REFERENCES	70
	INITIAL DISTRIBUTION LIST	72

LIST OF FIGURES

Figure 1.	The Model G Master-Slave Manipulator. . .	6
Figure 2.	Placement of Coordinate Frames.	9
Figure 3.	Flowchart of Calibration Process.	13
Figure 4.	Flowchart of Identification Process. . .	15
Figure 5.	Simple 2-Link Manipulator.	17
Figure 6.	Denavit-Hartenberg Model of Master-Slave Manipulator (Unconstrained Method)	19
Figure 7.	Programs Used in Unconstrained Calibration Process	25
Figure 8.	Denavit-Hartenberg Model of the Master-Slave Manipulator (Ballbar Method)	27
Figure 9.	Programs Used in Constrained (Ballbar) Calibration Process	29
Figure 10.	Kinematics of Ballbar	31
Figure 11.	Accuracy of Parameter Identification/Low Noise (Unconstrained Method).	35
Figure 12.	Manipulator Accuracy/Low Noise (Unconstrained Method).	35
Figure 13.	Accuracy of Parameter Identification/Low Noise (Ballbar Method).	36
Figure 14.	Manipulator Accuracy/Low Noise (Ballbar Method).	36
Figure 15.	Accuracy of Parameter Identification/Noise x 10 (Unconstrained Method).	37
Figure 16.	Manipulator Accuracy/Noise x 10 (Unconstrained Method).	37
Figure 17.	Accuracy of Parameter Identification/Noise x 10 (Ballbar Method).	38
Figure 18.	Manipulator Accuracy/Noise x 10 (Ballbar Method).	38

Figure 19.	Position Error at $x=1292.7$ mm.	42
Figure 20.	Position Error at $x=1542.7$ mm.	43
Figure 21.	Position Error at $x=1792.7$ mm.	44

LIST OF TABLES

TABLE I.	NOMINAL KINEMATIC PARAMETER TABLE (UNCONSTRAINED CASE)	21
TABLE II.	NOMINAL KINEMATIC PARAMETER TABLE (CONSTRAINED CASE)	28
TABLE III.	NOMINAL AND IDENTIFIED KINEMATIC PARAMETERS USING UNCONSTRAINED METHOD WITH LOW NOISE AND 60 OBSERVATIONS	39
TABLE IV.	NOMINAL AND IDENTIFIED KINEMATIC PARAMETERS USING BALLBAR METHOD WITH LOW NOISE AND 55 OBSERVATIONS	40

ACKNOWLEDGEMENTS

I would like to express my appreciation to my wife Rebecca and my children Charles, Joyce, and Rosa for the support and inspiration that they provided me. I would also like to extend a most sincere thanks to Professor Morris R. Driels for his patient guidance and technical expertise.

I. INTRODUCTION

The goal of using robot manipulators as the key link in flexible automated manufacturing systems has presented engineers with a variety of significant problems. For a six degree of freedom robot, the position and orientation of the manipulator end point must be specified for each pose. Accuracy and repeatability are the yardsticks of a robot's performance. Accuracy is the measure of the robot's ability to move to a commanded position in its workspace. Repeatability is the measure of the robot manipulator's ability to return to a previously learned position. Presently, robots that are used in industrial applications display adequate repeatability, but do not exhibit satisfactory levels of accuracy. For most industrial robots, repeatabilities of the order of 1 mm or better can be attained while the positioning accuracy may be off by as much as 1 cm [Ref. 1:p. 14]. For on-line programming applications such as the traditional automated pick and place operations where the robot manipulator must be taught the desired motion, adequate repeatability alone is sufficient. However, as the concept of off-line programming was developed as a means of automatically generating robot control programs for tedious applications that previously would have involved large numbers of taught tasks, the low levels of accuracy that

robot manipulators could attain became a major roadblock to their widespread usage.

There are several factors that adversely influence the accuracy of robot manipulators. Among them are: temperature variations, gear backlash and harmonics, compliance in links and joints, steady state errors in the joint servo controllers, and inaccurate knowledge of the manipulator's kinematic parameters. Experience has shown that the most prevalent source of error is inaccurate knowledge of the kinematic parameters that the robot controller has of the manipulator arm. This work deals primarily with the identification of the variations in the kinematic parameters of the model that the robot controller has.

Even small variations in these kinematic parameters can cause significant error in the manipulator end point placement. The calibration process identifies the actual kinematic parameters of the model and uses them to update the robot controller's model so that the manipulator end point may be placed into a commanded position with greater accuracy. In calibration tests performed by Mooring, Roth, and Driels [Ref. 2:p. 6] and several others, it has been shown that correction of the kinematic errors resulted in improvement in accuracy to the same order of magnitude as the repeatability.

The process of robot manipulator calibration is characterized by four major steps: modelling, measurement, identification, and correction. The first step in the

calibration process is to form a valid kinematic model of the manipulator. The model is the fundamental relationship between the manipulator's kinematic parameters and the resulting end effector pose. The manipulator model may take two basic forms. The forward kinematic model is used to compute the end effector pose given the joint variable data. The inverse kinematic model is used to determine the joint displacements for a given pose. The kinematic model is constructed using the Denavit-Hartenberg method with modifications. The resulting model is used to define an error quantity based on the nominal kinematic parameter set and the unknown actual kinematic parameter that need to be identified.

Measurement involved physically moving the manipulator end effector to various locations in its workspace and recording the corresponding joint displacements. There are a number of methods that have been used to obtain the data necessary for manipulator calibration. Theodolites [Ref. 3], laser interferometers [Ref. 4], coordinate measuring machines [Ref. 5], and many other techniques can be used depending on the constraints imposed by the desired level of accuracy, size, ease of use, and cost. Alternatively, joint variable data and pose information can be obtained through computer simulation with the use of a random number generator routine. This was the approach employed in this research.

In the identification phase, the task is to identify the set of model parameters that allow the poses computed from the

model to most closely match the measured data. This is accomplished through the use of a gradient based Levenberg-Marquardt algorithm that used the collected pose information to identify the actual parameters by systematically changing the nominal parameters to reduce the previously defined error quantity. There are several factors that influence the identification process. These factors are the type of identification routine used, the initial values of the parameters to be determined, the number of poses taken, the influence of measurement accuracy and noise, encoder noise, the choice of measurement configuration, and the attainable range of joint displacements used during the observations. These effects are discussed in detail at a later time.

Finally, in the correction step, these identified kinematic parameters are used to update the robot controller's model. This process, however, is not without its own unique set of problems. Normally, an inverse kinematic solution using the actual kinematic parameters is employed to convert the desired off-line locations in the task space to modified locations in the manipulator's own joint space. The robot has an inverse kinematic solution for the nominal model, but may have to develop its own solution using the actual model. These issues are beyond the scope of this research and were not addressed.

The purpose of this research was to compare the accuracy attained for two different computer simulated calibration

methods. The first method involves using an unconstrained manipulator end point and the second method employs a passive end point motion constraint called a ballbar. These computer simulated calibrations were performed on the Model G Compact Master-Slave Manipulator shown in Figure 1. The Model G Master-Slave Manipulator is a six degree of freedom manipulator arm with five revolute joints and one prismatic joint (5R1P). This manipulator is designed to reproduce the natural movements and force of the human hand. The manipulator end point will move in exactly the same manner as the operator moves the manipulator handle. The motion is constrained only by the dimensional limits of the manipulator itself. The forces produced at the end point will be the same as those forces applied at the handle with the exception of minor losses due to unbalance and friction. This manipulator was chosen for these calibrations because of its usefulness for experiments that are concerned with probing of objects that can not be viewed during the probing operation to acquire contact information.

The format of the remainder of this thesis will be to first conduct an in-depth examination of theory applicable to robot calibration. This will be followed by an analysis of the two calibration methods used and the unique problems with each method. Next, will be a discussion of the results obtained and, finally, the conclusions drawn from this research will be stated.

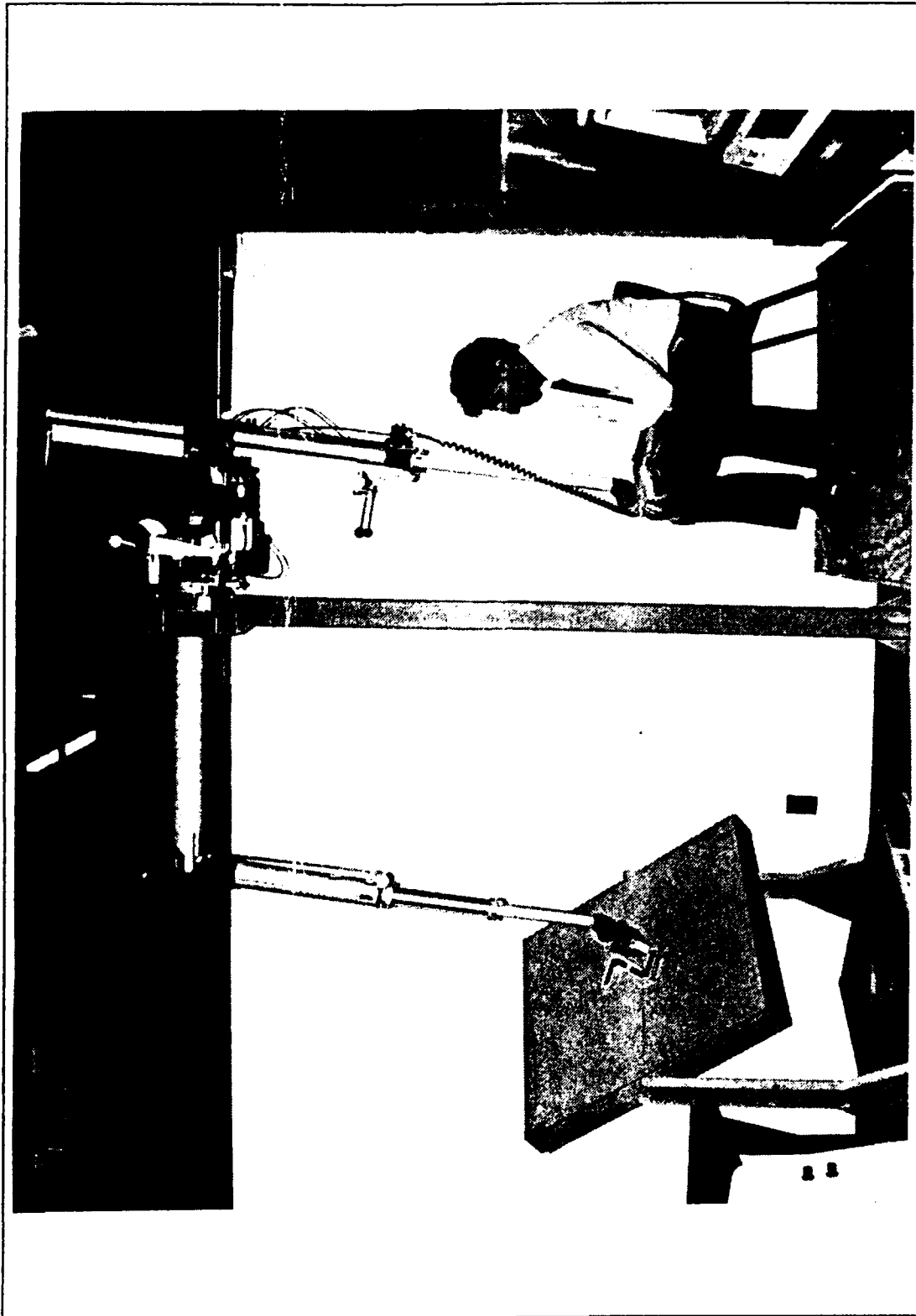


Figure 1. The Model G Master-Slave Manipulator.

II. THEORY

A. THE DENAVIT-HARTENBERG METHOD WITH MODIFICATIONS

As was discussed in the Introduction, the starting point for any calibration process is the establishment of a representative model of the manipulator. There are currently a number of methods of generating the forward kinematic model of a serial link manipulator. The technique that was used to define the spatial orientation between objects and various locations in the manipulator working volume is the Denavit-Hartenberg method [Ref. 6] with modifications proposed by Hayati [Ref. 7], Mooring [Ref. 8], and Wu [Ref. 9] to handle situations in which consecutive joint axes are nominally parallel. The basic concept is to place a coordinate frame on each of the manipulator links using a set of rules that defines the origin of the frame and its orientation. The position of consecutive links is specified by a homogeneous transformation matrix, which transforms the frame fixed on link $n-1$ into the frame fixed on link n . This transformation is composed of more fundamental transformations representing three basic translations along the x , y , and z axes and three rotations about those same axes. These 4×4 matrix transformations are expressed as follows:

$$Trans(x, y, z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

$$ROT(x, \theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$ROT(y, \theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$ROT(z, \theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

where $trans(x, y, z)$ describes a translation given by the vector $r = [x, y, z]^T$ and $ROT(x, \theta_x)$ describes a rotation of θ_x about the x-axis of the coordinate frame.

With the aid of Figure 2, the Denavit-Hartenberg transformation methodology can be illustrated. First, the axis of joint motion must be identified and the z-axis must be aligned with the axis of joint motion. Next, the common normal between consecutive joint axes must be identified.

Then, the origin of coordinate frame n is located at the intersection of joint axis $n+1$ and the common normal between

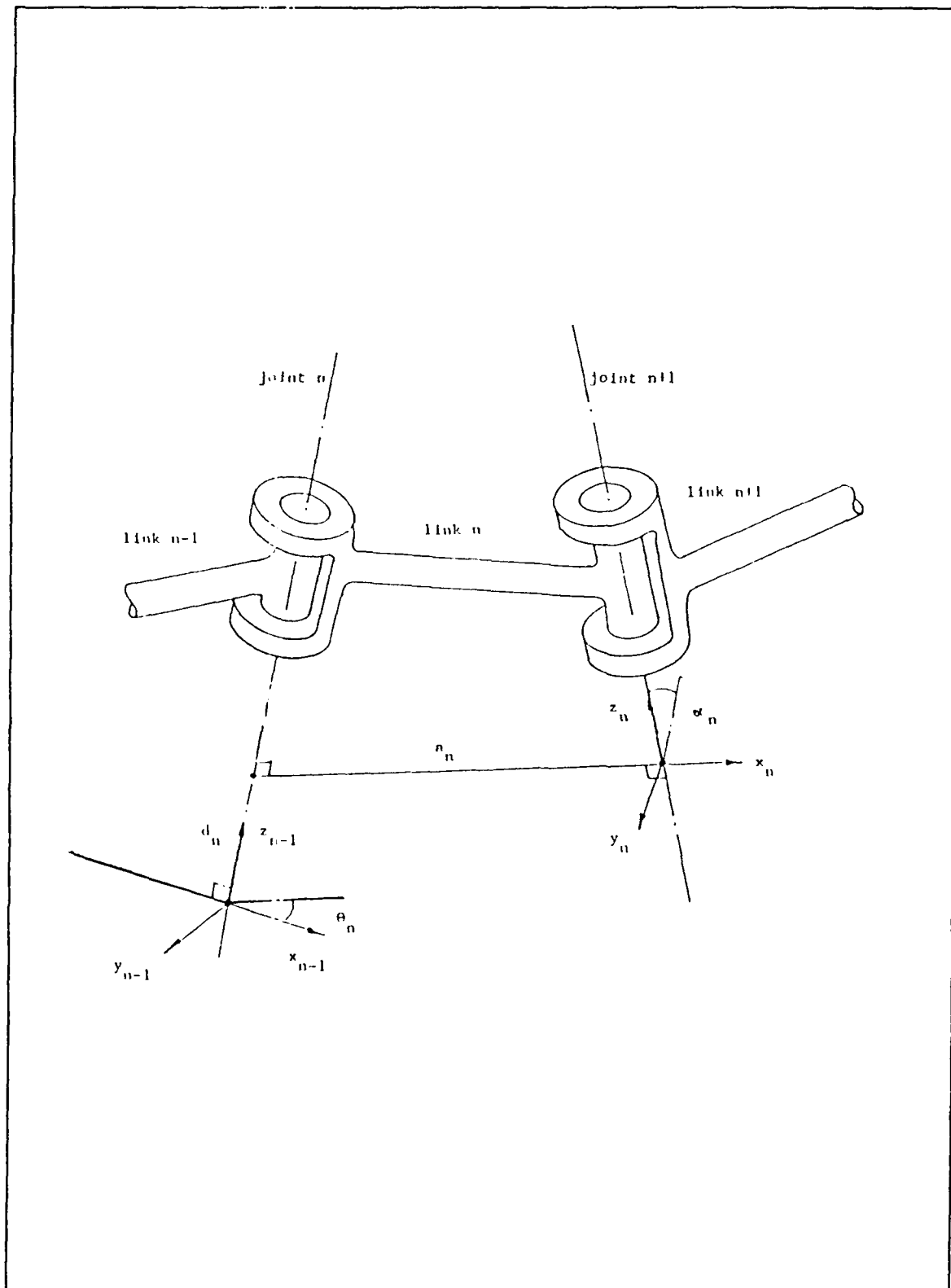


Figure 2. Placement of Coordinate Frames.

axes $n+1$ and n . the z axis of coordinate frame n is always aligned with joint axis $n+1$ and the x axis is always aligned along the common normal between consecutive joint axes.

Transforming frame $n-1$ to frame n is accomplished by the following sequential steps:

- Rotate frame $n-1$ about axis z_{n-1} by an angle θ_n , the joint angle.
- Translate along axis z_{n-1} a distance d_n , the offset.
- Translate along the rotated x_{n-1} axis, a distance a_n , the link length.
- Rotate about axis x_n by an angle α_n , the twist angle.
- Rotate about axis y_n by an angle β_n .

Incorporating these rules with the transformation matrix format specified in Equations 1, 2 and 3, the transformation from frame $n-1$ to frame n is expressed in the following form:

$$A_n = ROT(z, \theta_n) Trans(z, d_n) Trans(x, a_n) ROT(x, \alpha_n) ROT(y, \beta_n) \quad (5)$$

Performing the matrix multiplications gives the resulting form:

$$A_n = \begin{bmatrix} c\theta_n c\beta_n - s\theta_n s\alpha_n s\beta_n & -s\theta_n c\alpha_n & c\theta_n s\beta_n + s\theta_n s\alpha_n c\beta_n & a_n c\theta_n \\ s\theta_n c\beta_n + c\theta_n s\alpha_n s\beta_n & c\theta_n c\alpha_n & s\theta_n s\beta_n - c\theta_n s\alpha_n c\beta_n & a_n s\theta_n \\ -c\alpha_n s\beta_n & s\alpha_n & c\alpha_n c\beta_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

In most cases, four out of the five transformations are necessary to transform frame $n-1$ into frame n . For revolute joints, the parameters d_n , a_n and α_n are constants dictated by the geometry of the manipulator and θ_n is the angular joint

variable. The parameter β is defined only in cases where consecutive coordinate axes are parallel and, in these instances, d_n , is normally set to zero. When consecutive axes are parallel, there is no unique common normal. The β_n parameter allows for small amount of inclination between the axes. For prismatic joints, the location of the origin of the coordinate system is determined by extending the axis so that it intersects the axis of the next joint. This makes the length of the common normal, a_n , and the next joint offset, d_{n+1} , both equal to zero. Therefore, for prismatic joints, d_n is the joint variable and the link geometry is described by θ_n and α_n .

In order for a robot manipulator to have complete dexterity in its working volume, it must have six degrees of freedom. For a six joint six link manipulator, the transformation from frame 5 to frame 6 takes the form

$$A_6 = ROT(z, \phi_6) ROT(y, \theta_6) ROT(x, \psi_6) Trans(p_{x6}, p_{y6}, p_{z6}) \quad (7)$$

where the rotations are sequentially defined as roll, pitch and yaw [Ref. 10]. The transformation from the base coordinate frame to the manipulator end link is:

$$T_6 = A_1 A_2 A_3 A_4 A_5 A_6 \quad (8)$$

Any suitable calibration model must be, in Everett's terms [Ref 11], complete. Completeness refers to the model's ability to relate joint displacements to the tool pose for a manipulator while allowing for the arbitrary placement of the

world coordinate frame and arbitrary assignment of the manipulator's zero position. In other words, the model must have the proper number of identifiable parameters to account for variations in those parameters. The required number of independent kinematic parameters is the same as the number of constraint equations necessary to specify the tool pose and joint frames. Mooring, Roth and Driels [Ref. 2:p. 43] have concluded that for each revolute joint, four independent kinematic parameters are needed and for each prismatic joint, two independent kinematic parameters are required. The required number of independent kinematic parameters N , can be determined from the following equation.

$$N = 4R + 2P + 6 \quad (9)$$

R is the number of revolute joints and P is the number of prismatic joints. An additional six parameters are specified in order to obtain an independent tool frame location.

B. PARAMETER IDENTIFICATIONS

The flowchart in Figure 3 outlines the calibration process up through the identification step. First, the range of motion for each joint and the number of observations to be made must be determined. Next, sets of joint variables for each observations are obtained with the use of a random number generator program. The forward kinematic model of the 5R1P manipulator is then applied to generate pose data for each of

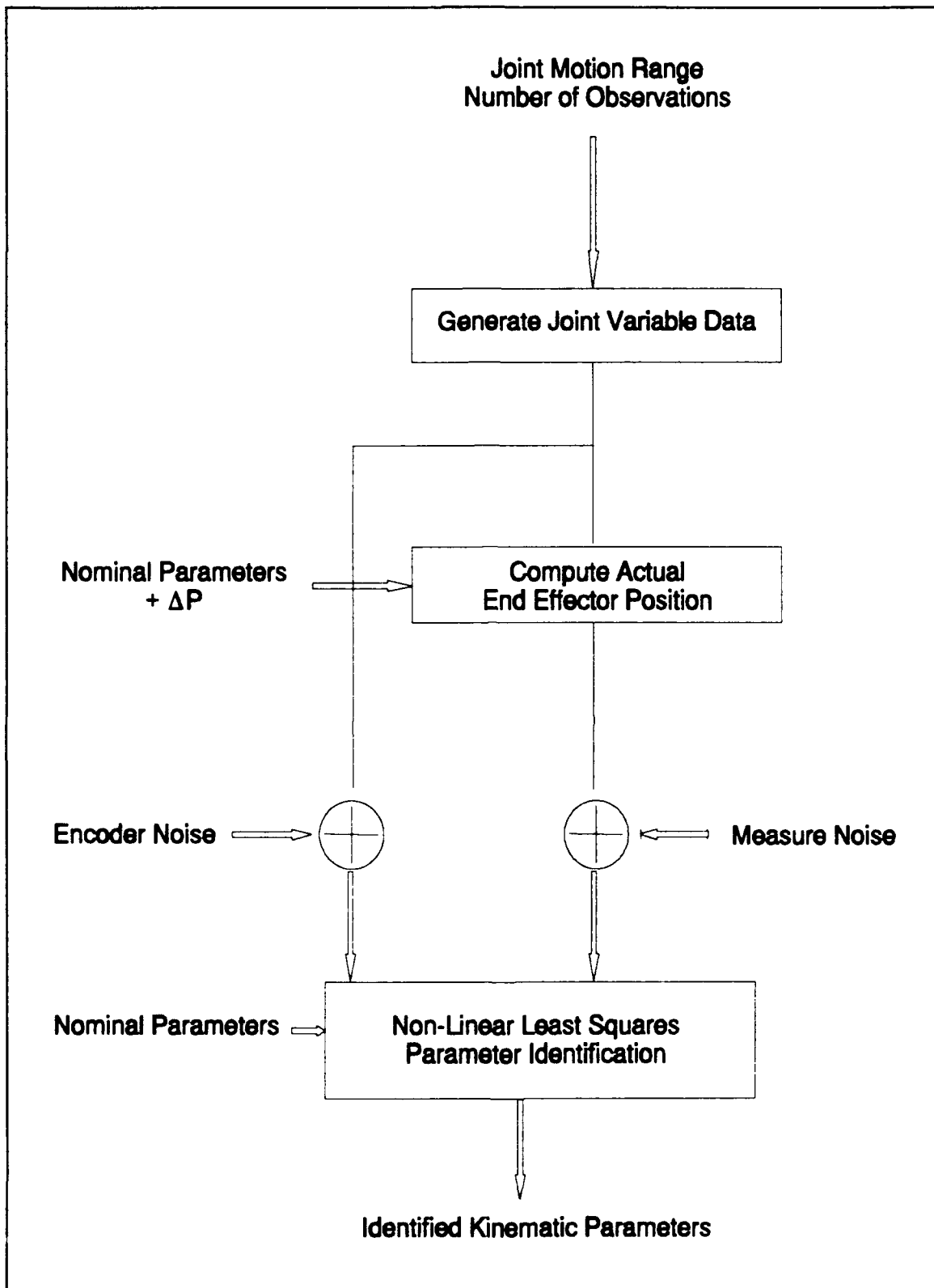


Figure 3. Flowchart of Calibration Process.

the observations using the joint variable data and manipulator link parameters. The kinematic parameters used in this forward kinematic solution are the nominal kinematic parameters plus a known error parameter, ΔP . The error parameter can take the form of a length error or an angular error as is appropriate for each of the parameters that is to be identified. The result of this application is a simulated known manipulator pose for each of the joint variable sets. The random noise of measurement and encoder noise were superimposed on the pose data and joint variables respectively.

The simulated observation data and the nominal kinematic parameters are the inputs to an identification program ID6. ID6 initializes the nominal kinematic parameters and feeds them to an identification subroutine ZXSSQ which numerically estimates the gradient and uses it to produce improved predictions of the kinematic model parameters. ZXSSQ employs a subroutine that takes the current parameter estimates and calculates an error between the model predictions and the simulated observation data. ZXSSQ uses the error to determine the gradient. The cycle continues until convergence criteria is met. The parameter estimation is treated as an unconstrained non-linear optimization problem. Figure 4 shows a flowchart of the process.

ZXSSQ is a finite difference, Levenberg-Marquardt routine that is tailored for non-linear least squares problems [Ref. 2:pp. 135-139]. The best way to illustrate its use is through

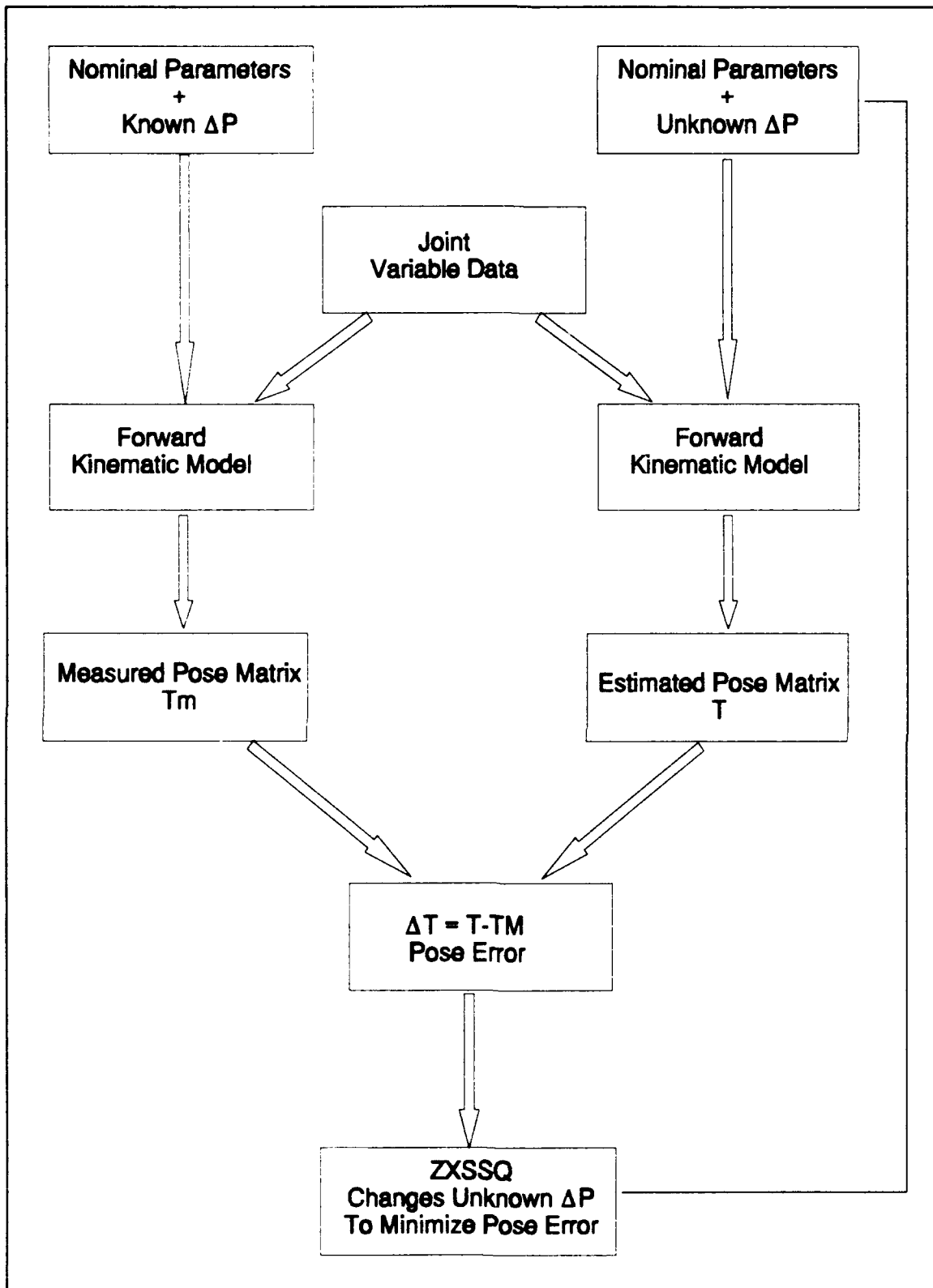


Figure 4. Flowchart of Identification Process.

an example. Consider the simple two link manipulator in Figure 5. What values of θ_1 and θ_2 will put the manipulator end point at point Q? If $P=F(\theta_1, \theta_2)$, ℓ_1 and ℓ_2 are known link lengths, and Q is a known position in the two dimensional workspace, the problem is to find θ_1 and θ_2 such that the quantity $z=(P-Q)$ approaches zero. In two-dimensional matrix notion, the equations are as follows:

$$z = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} P_x - Q_x \\ P_y - Q_y \end{bmatrix} = \begin{bmatrix} \ell_1 \cos \theta_1 + \ell_2 \cos (\theta_1 + \theta_2) - Q_x \\ \ell_1 \sin \theta_1 + \ell_2 \sin (\theta_1 + \theta_2) - Q_y \end{bmatrix} \quad (10)$$

Where x_e and y_e are the difference between the estimated and known parameters in the x and y directions respectively. The Levenberg-Marquardt algorithm uses this error quantity to numerically estimate the gradient and produces updated θ_1 and θ_2 values. The process continues until predetermined convergence criteria are met and θ_1 and θ_2 will be the values needed to put the manipulator end point at point Q. A more detailed explanation of the ZXSSQ algorithm is given in Appendix A.

According to Equation 9 for the 5R1P manipulator used in this study, it would be expected that 28 kinematic parameters would have to be identified. It will later be seen that this number will have to be altered to meet the particular needs of the measurement method employed and to achieve satisfactory parameter identification.

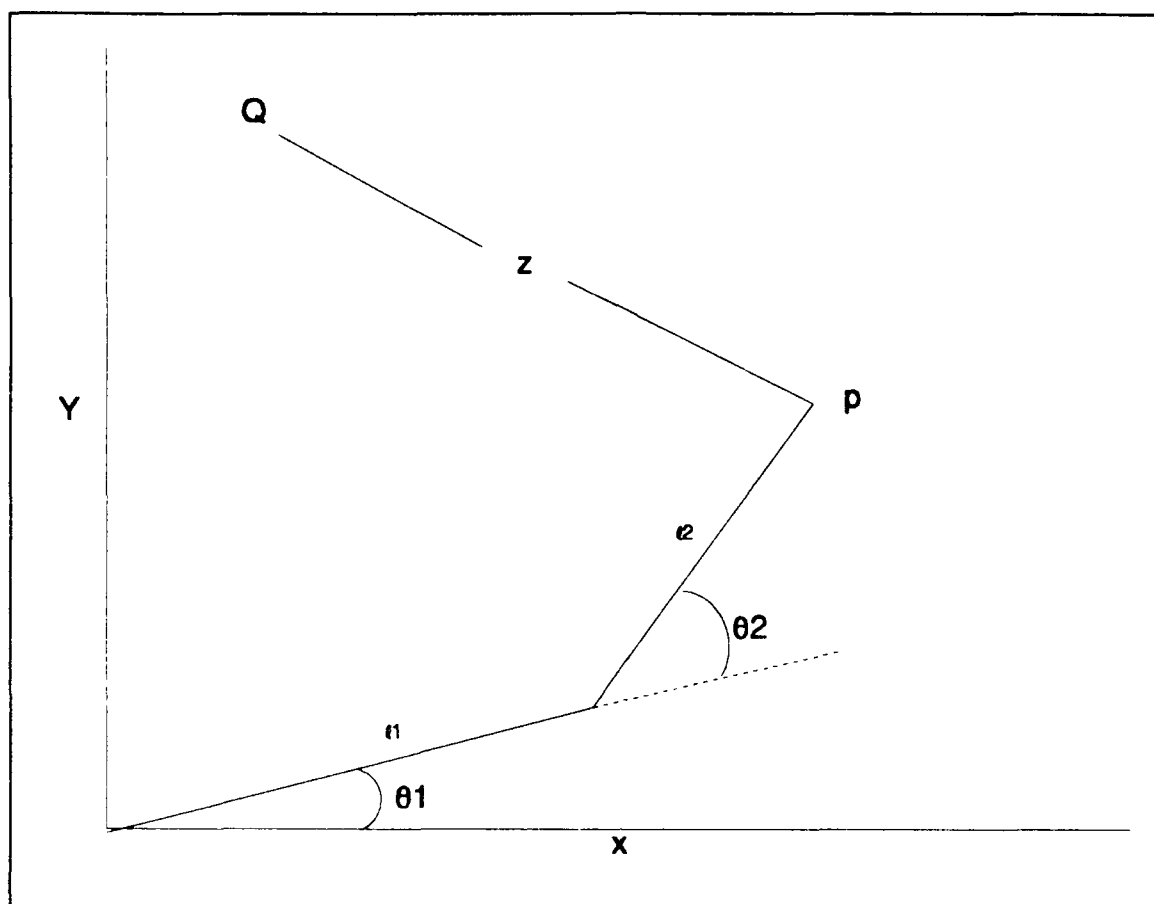


Figure 5. Simple 2-Link Manipulator.

III. CALIBRATION METHODS

A. UNCONSTRAINED METHOD

This method involves establishing a valid forward kinematic model using the Denavit-Hartenberg methodology and using this model to generate the manipulator poses that will be input to the identification program. Employing the Denavit-Hartenberg criteria previously discussed to the Model G Master-Slave Manipulator produces the model shown in Figure 6. The location of the world coordinate is arbitrary and it was fixed in the position shown only because it was a convenient reference frame for measurements. The location of the base frame of the manipulator is also arbitrary as long as Z_0 is aligned with the first joint axis. The remainder of the coordinate frames were allocated in accordance with the Denavit-Hartenberg Method. The transformation from frame 2 to frame 3 required the use of the parameter β_3 because coordinate frames 2 and 3 are nominally parallel [Ref. 12]. The definition of frame 6 is arbitrary and, in general, the transformation from frame 5 to frame 6 requires three translations and three rotations. However, since frame 6 was chosen to be offset from the origin of coordinate frame 5, its orientation is undefined. Therefore, only three parameters are required to transform from frame 5 to frame 6 and at least one

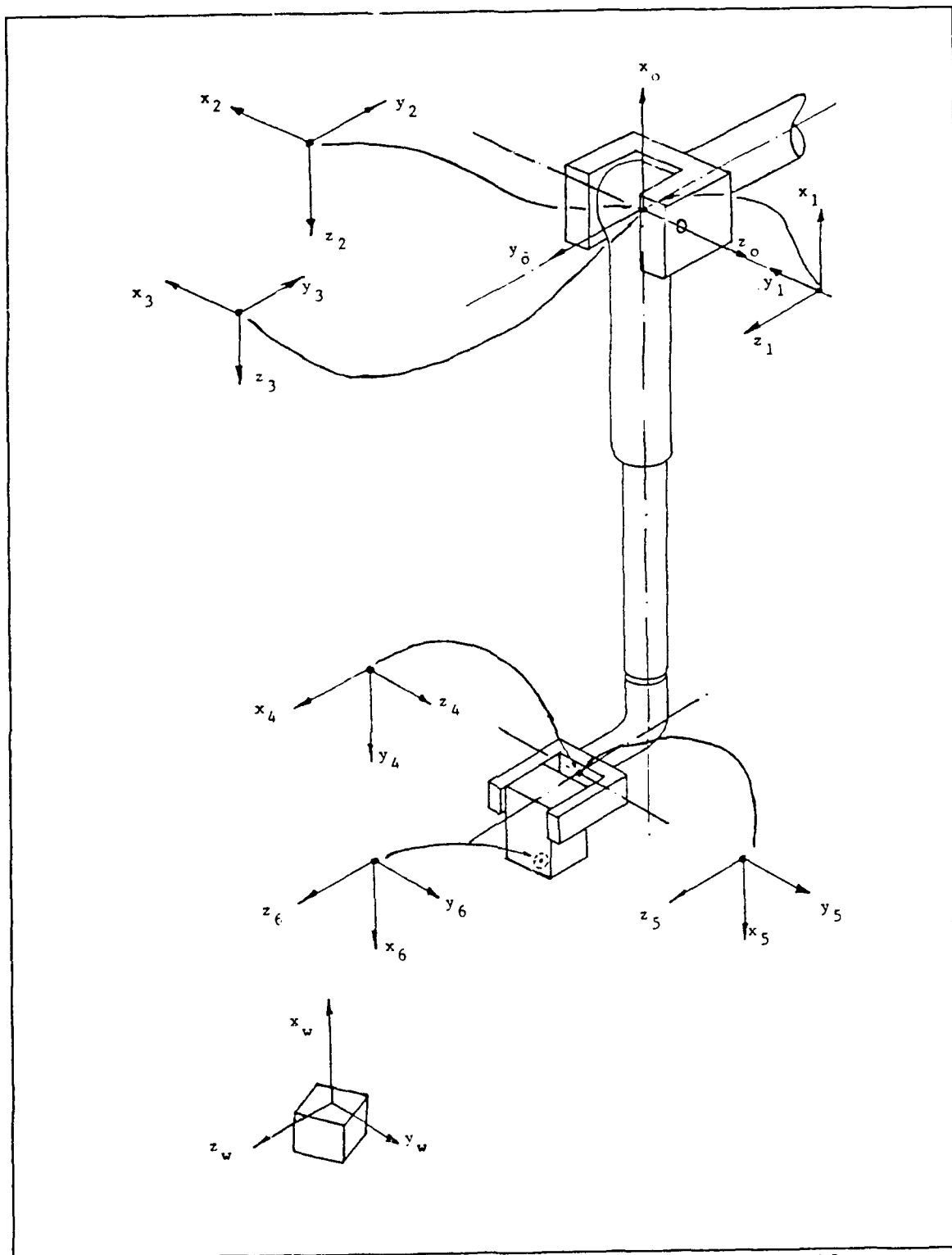


Figure 6. Denavit-Hartenberg Model of Master-Slave Manipulator (Unconstrained Method)

of these parameters must be a displacement otherwise no movement of the origin of frame 5 would occur.

The parameters ϕ_6 , p_{x_6} and p_{z_6} were chosen to define the transformation from frame 5 to frame 6 because the identified deviation from the nominal value of ϕ_6 is composed of the encoder offset $\delta\theta_6$ and the constant offset $\delta\phi_6$ as well as the joint variable θ_6 . The definition of ϕ_6 is as follows:

$$\phi_6 = \theta_6 + \delta\theta_6 + \delta\phi_6 \quad (11)$$

The transformation from frame 5 to frame 6 can be expressed in a reduced form

$$A_6 = ROT(z, \phi_6) TRANS(p_{x_6}, 0, p_{z_6}) \quad (12)$$

As was discussed previously, in order for a calibration model to be valid, it must satisfy the completeness criteria specified in Equation 9. The required number of identifiable, independent kinematic parameters must equal the number of constraint equations needed to define the tool pose [Ref. 2.p. 42]. Using the completeness criteria, it is expected that there must be 28 identifiable kinematic parameters in order to have a valid model of a 5R1P manipulator. However, because only three parameters instead of six were used to specify the transformation from frame 5 to frame 6, the required number of independent kinematic parameters is 25 for this calibration process.

There were some unique problems encountered applying the Denavit-Hartenberg methodology to the Model G Master-Slave

Manipulator. The prismatic joint, in particular, requires some modifications to the Denavit-Hartenberg model. Because the location of frame 3 is defined by the common normal between axes 4 and 5 and the prismatic joint axis (frame 3) is not fixed in space, the prismatic joint axis is free to move and it moves through the origin of frame 3 [Ref. 1:p. 18]. Consequently, the parameters a_3 and d_4 are always zero. In addition, the parameter θ_3 must be set at a constant value because it cannot be identified independently from θ_4 for this manipulator configuration. Table I shows the table of the nominal kinematic parameters. The parameters that are defined to be zero are in boldface type.

**TABLE I. NOMINAL KINEMATIC PARAMETER TABLE
(UNCONSTRAINED CASE)**

	$\delta\theta_w$ DEG	d_w mm	a_w mm	α_w DEG	β_w DEG	
	0.0	0.0	1492.3	90.0	0.0	
Link #	$\delta\theta_i$ DEG	d_i mm	a_i mm	α_i DEG	β_i DEG	
1	0.0	0.0	0.0	-90.0	0.0	
2	90.0	0.0	0.0	-90.0	0.0	
3	0.0	730.3	0.0	0.0	0.0	
4	-90.0	0.0	82.55	90.0	0.0	
5	90.0	0.0	0.0	90.0	0.0	
	$\delta\phi_e$ DEG	θ_e DEG	ψ_e DEG	px_e mm	py_e mm	pz_e mm
	0.0	0.0	0.0	50.8	0.0	50.8

Having established a valid, working model of the Model G Master-slave Manipulator, the task was then to obtain joint variable data for a variety of manipulator poses. This was

accomplished using Program JOINT. Program JOINT uses a random number generator subroutine to generate the joint variable data and then stores it in a data file called TELE-VAR.DAT. Program JOINT is run a second time to obtain joint variable data that will be used in a verification program that will be described in more detail later. This second set of joint variable data is stored in file POSEVER.DAT.

The next step is to generate pose information for the Model G Manipulator simulation. Program POSE reads the joint variable data from file TELE-VAR.DAT and the table of nominal kinematic parameters from file INPUT.DAT and computes the manipulator pose using a forward kinematic solution. The set of joint variables and the corresponding manipulator end point pose information are stored in file TELE-POS.DAT. In program POSE, estimates of measurement noise and the encoder offsets are added to the data through INPUT.DAT.

The actual kinematic parameters are identified by program ID6, using the previously discussed non-linear least squares method. Program ID6 consists of three main components. The first component is where the nominal kinematic parameters are read from INPUT.DAT and the pose data for each observation are read from TELE-POSE.DAT. The program then defines the initial values of the model and the parameters required by the identification subroutine ZXSSQ are initialized. The identification subroutine ZXSSQ is the second major component of program ID6. The details of how ZXSSQ works can be found in

the parameter identification section of Chapter II and in Appendix A. ZXSSQ iteratively estimates the gradient and uses the estimate to produce an updated approximation of the model parameters. The cycle continues until the kinematic parameters are identified consistently to four significant figures. The third major component of program ID6 is the subroutine TELE-ARM which takes the current estimate of the model parameters, computes a forward kinematic solution using the estimated parameters, and then calculates the error between the model prediction and the measured pose data. This error is, in turn, used by ZXSSQ to determine the gradient. The output of program ID6 is file RESULT.DAT which consists of the actual, identified kinematic parameters of the manipulator and the calculated RMS difference between the nominal and identified kinematic parameters. The RMS quantity is broken down into length and angular error parameters (K_L and K_A) respectively. These error parameters reflect the accuracy of the identification process.

The final stage of the computer simulated calibration process is a verification program designed to determine the accuracy that the manipulator could attain if the identified kinematic parameters were to replace the nominal parameters in the manipulator's controller. Program VERIFY reads the nominal kinematic parameters from INPUT.DAT and the identified parameters from RESULT.DAT and computes separately for each set of parameters a forward kinematic solution. These

solutions are used to calculate the differential position and orientation matrix.

$$\Delta T = \begin{bmatrix} 0 & -\delta_z & \delta_y & dx \\ \delta_z & 0 & -\delta_x & dy \\ -\delta_y & \delta_x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

The position error is calculated as follows:

$$POSERR = \sqrt{dx^2 + dy^2 + dz^2} \quad (14)$$

The position error is indicative of the accuracy of the calibrated manipulator. Figure 7 shows a flowchart of the programs used in the simulated calibration process. These programs and data files can be reviewed in Appendix B.

B. CONSTRAINED (BALLBAR) METHOD

The ballbar method involves the use of a passive end point motion constraint to obtain pose data. The end point of the manipulator is connected to a fixed point on a table by means of a ballbar of known length. Figure 8 shows the Model G Master-Slave Manipulator configuration with the ballbar of length 552.8 mm attached. This ballbar length was obtained by fixing the ballbar at a location very near the manipulator end point when it is in the zero position. The other end of the bar was connected to the manipulator end point. The ballbar was then cycled through its reachable volume while constrained by the dimensional limits of the manipulator. This approximate

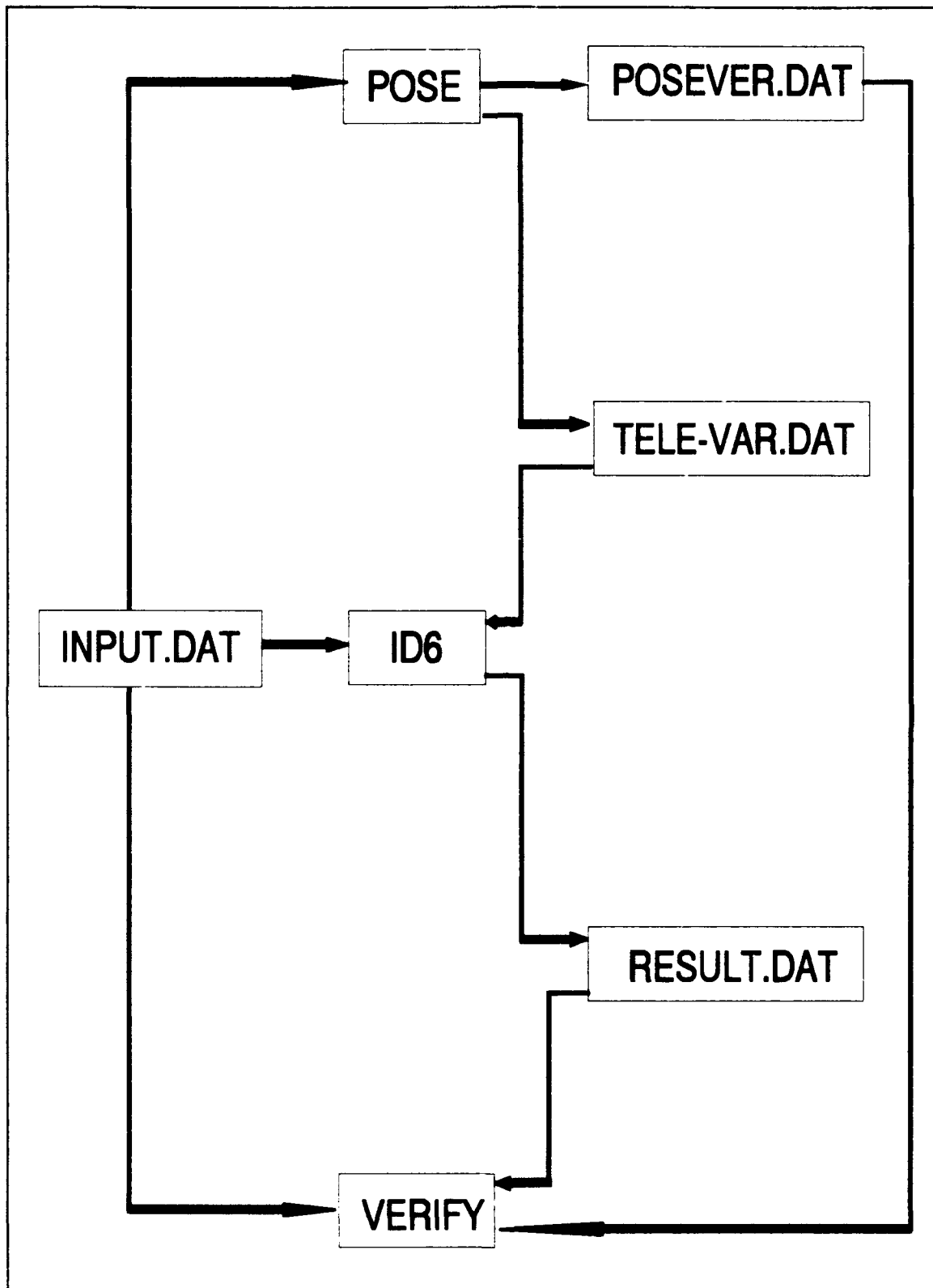


Figure 7. Programs Used in Unconstrained Calibration Process

bar length was chosen because it maximized the possible range of joint displacements.

The coordinate frames were allocated in accordance with the Denavit-Hartenberg criteria. With the exception of the world coordinate frame, the coordinate frames were positioned exactly as they were for the unconstrained calibration. However, special consideration must be given to the choice of world coordinate frame. If it is assumed that all the Model G Manipulator joint displacements are fixed, a rotation of frame 0 relative to the world frame results in a change of the manipulator end point (the origin of frame 6) coordinates in the x_w , y_w , z_w frame. The distance between the world coordinate frame and manipulator end point is the fixed length of the ballbar and it remains unchanged. Since the relative rotation between the world frame and the base frame cannot be measured the conclusion is that this rotation cannot be identified. Hence, it is logical that the world coordinate frame selected be orthogonal to the base frame. The location of the base frame is arbitrary as long as the z_0 axis is aligned with the first joint axis.

The transformation from the world coordinate frame to the base frame is a function of the parameters x_w , y_w , and z_w only. Because parameters z_w and d_1 are measured in the same direction, they cannot both be identified. Therefore, the parameter z_w is set to zero. The transformation from the world coordinate frame to the base frame can be expressed as follows

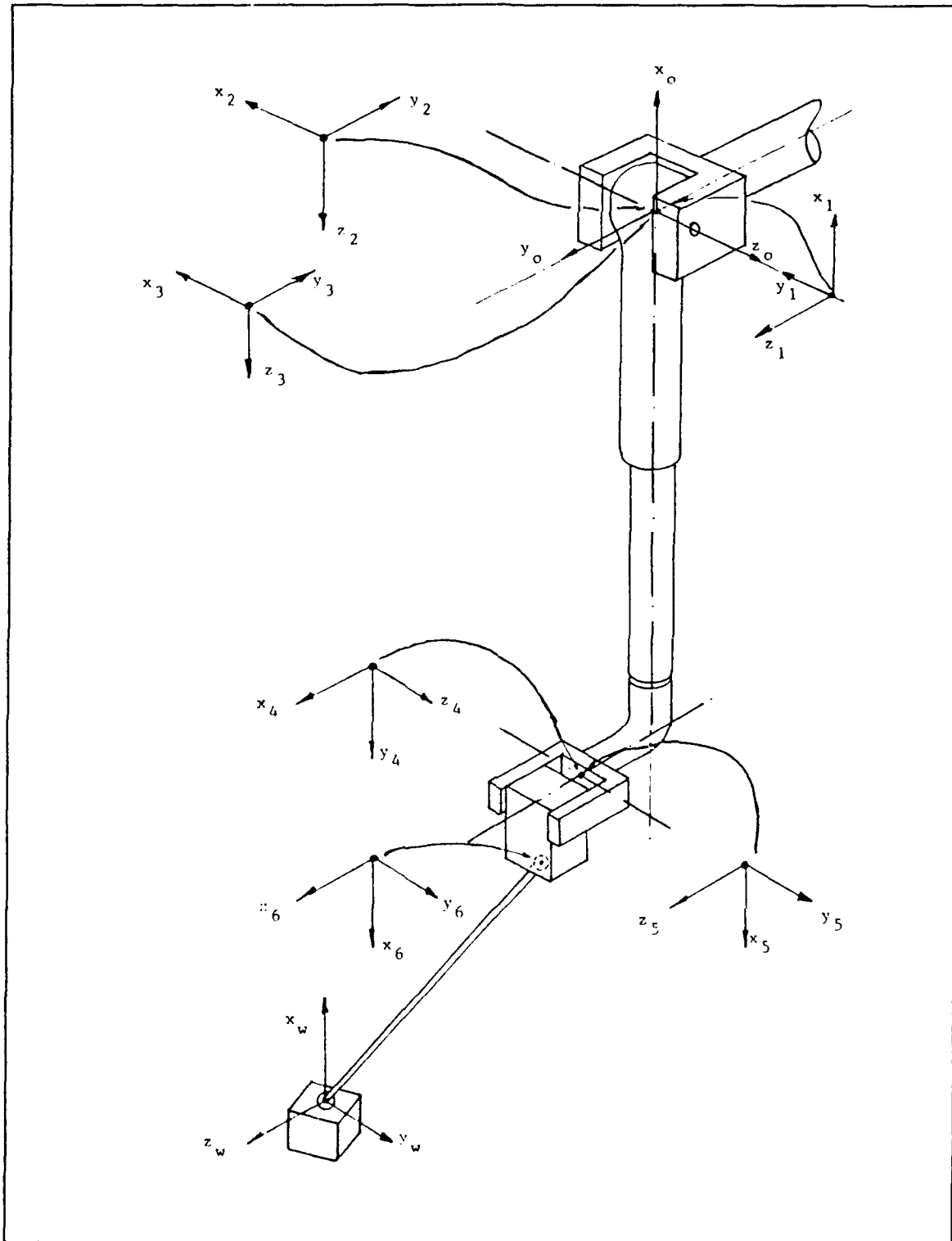


Figure 8. Denavit-Hartenberg Model of the Master-Slave Manipulator (Ballbar Method)

$$A_c = TRANS (x_w, y_w, 0) \quad (15)$$

In addition, it has been determined the parameters x_w , y_w , and $\delta\theta_1$ are not independent. The parameter $\delta\theta_1$ cannot be identified and was set to zero. The table of nominal kinematic parameters for ball method calibration of the Model G Manipulator is shown in Table II. The parameters in bold face type were not identified in the calibration process. For the ballbar calibration method, there are 22 kinematic parameters that must be identified.

**TABLE II. NOMINAL KINEMATIC PARAMETER TABLE
(CONSTRAINED CASE)**

		x_w mm	y_w mm	z_w mm	
		1542.7	-132.95	0.0	
Link #	$\delta\theta$	d_i	a_i	α_i	β_i
1	0.0	0.0	0.0	-90.0	0.0
2	90.0	0.0	0.0	-90.0	0.0
3	0.0	0.0	0.0	0.00	0.0
4	-90.0	0.0	82.55	90.0	0.0
5	90.0	0.0	0.0	90.0	0.0
$\delta\phi_e$ DEG	θ DEG	ψ DEG	px_e mm	py_e mm	pz_e mm
0.0	0.0	0.0	50.8	0.0	50.8

As in the unconstrained calibration process, the simulated ballbar method calibration of the Model G Manipulator is accomplished with a series of computer programs. A flowchart of the programs used is shown in Figure 9. The ballbar length

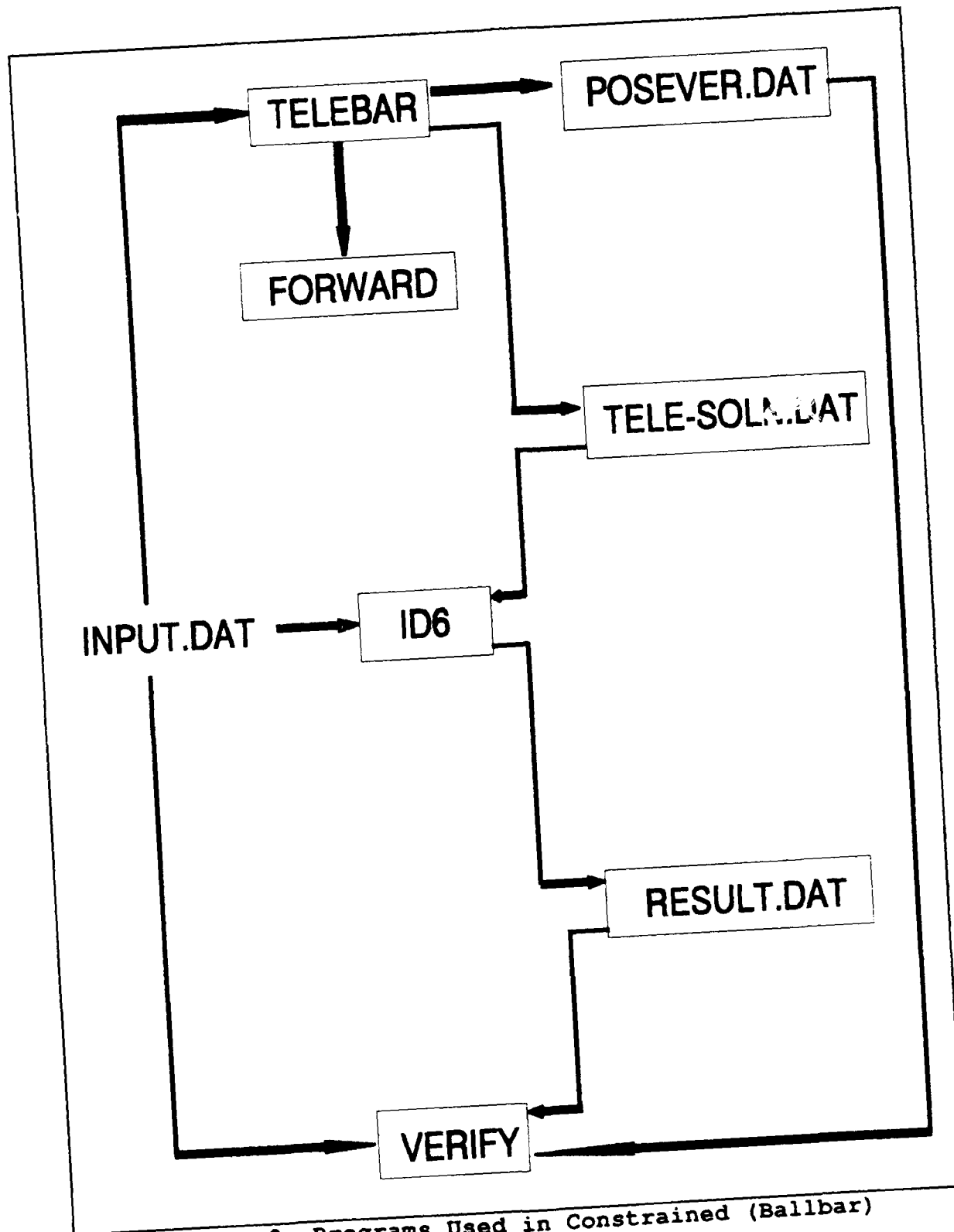


Figure 9. Programs Used in Constrained (Ballbar) Calibration Process

is fixed at 552.8 mm and at each end of the bar is a ball joint capable of 90° of solid angle rotation. One end of the bar is attached to a fixed point in space and the other end is attached to the end flange of the Model G Manipulator. Pose information is created in program TELEBAR by using a random number generator routine to generate the angles that the bar makes with the z , and y , axes (Figure 9). The position of the ballbar end point is

$$[x, y, z]^T = R [0, 0, 0, 1]^T \quad (16)$$

where the transformation R is defined as follows

$$R = ROT(z, \theta) ROT(y, \phi) TRANS(x, r) \quad (17)$$

For each pose, the distance d from the manipulator end point to the origin of the world coordinate frame is calculated using

$$d = \sqrt{x^2 + y^2 + z^2} \quad (18)$$

where x , y , and z are the coordinates of the manipulator end point in space relative to the world coordinate frame. Program TELEBAR employs the previously discussed non-linear least squares algorithm ZXSSQ to minimize the function

$$F_i = |d_i - \ell| \quad (19)$$

where ℓ is the length of the ballbar subroutine ZXSSQ iteratively estimates a gradient and produces an approximation of the joint displacements necessary to establish the current

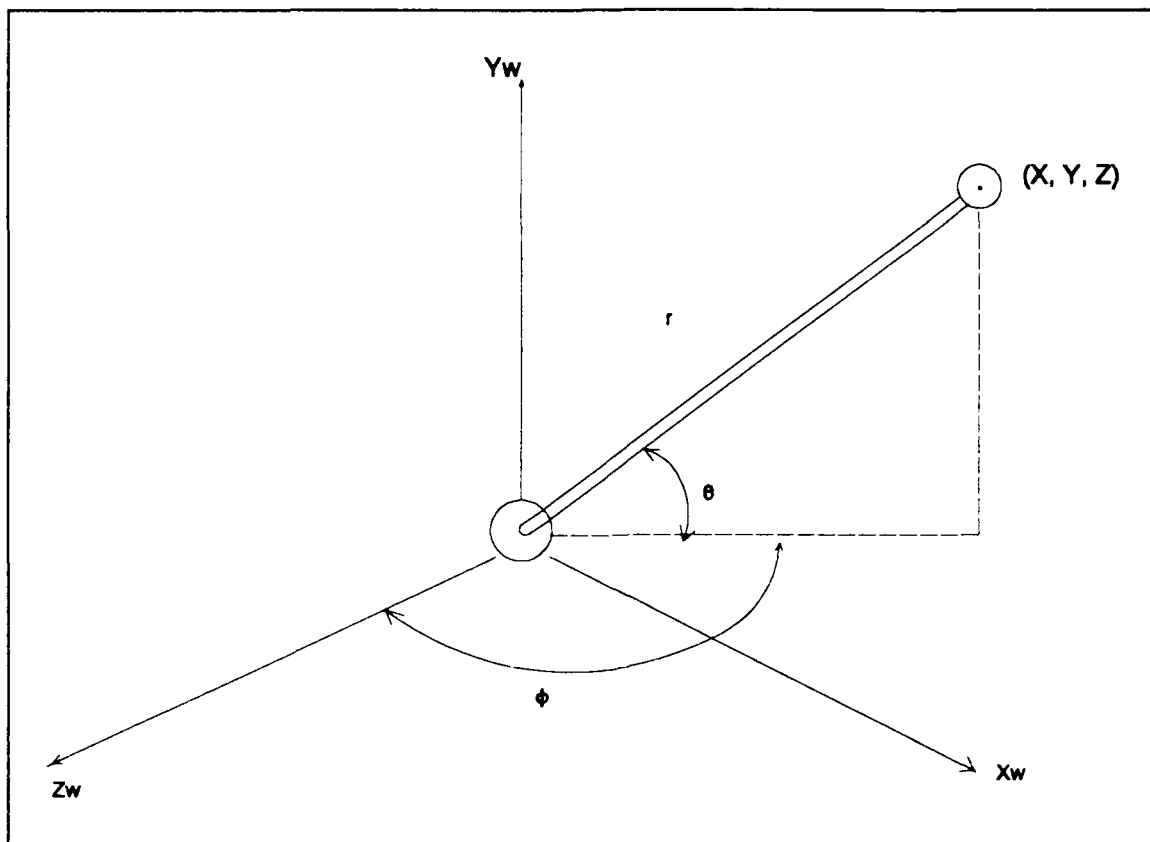


Figure 10. Kinematics of Ballbar

pose. ZXSSQ uses subroutine TELE-ARM to compute the forward kinematic solution of the manipulator using the current values of joint displacements. TELE-ARM calculates F_j which is used by ZXSSQ to determine the gradient. The cycle continues until the joint variables for the pose are consistently identified to four significant figures. The joint displacements are stored in file TELE-SOLN.DAT. Program TELEBAR is run a second time to generate joint variable displacements for use in the verification phase. The second set of joint variables is stored in file POSEVER.DAT.

Program FORWARD is used to check the validity of the joint variable data generated by program TELEBAR. Program FORWARD

computes the forward kinematic solution for the Model G Manipulator for each set of joint displacements. The distance d from the manipulator end point to the world coordinate frame is

$$d = \sqrt{x^2 + y^2 + z^2} \quad (19)$$

where x , y , and z are the coordinates of the manipulator end point relative to the world coordinate frame. If d equals the length of the ballbar, the corresponding set of joint displacements is valid. Program FORWARD is not a part of the calibration process, but it is an expeditious way to check the joint variable data.

As in the unconstrained calibration process, the identification of the actual kinematic parameters of the manipulator is accomplished by program ID6. Program ID6 reads the nominal kinematic parameters from INPUT.DAT and the pose information from TELE-SOLN.DAT. The actual kinematic parameters of the manipulator are stored in file RESULT.DAT. Program VERIFY calculates the accuracy of the manipulator using the actual identified kinematic parameters as was explained earlier. These computer programs are shown in Appendix B.

IV. DISCUSSION

In order to obtain a satisfactory comparison of the two calibration methods, a number of computer simulated calibrations were performed on each configuration. In these simulations, the independent variables were the number of observations taken and the level of measurement noise present. The dependent variables were the accuracy of parameter identification and the manipulator accuracy using the identified kinematic parameters. The results are shown in Figure 11 through Figure 18. For the unconstrained configuration, the low noise value was set at 0.1 mm. The accuracy of parameter identification and the position error were each separately plotted against the number of observations. The same procedure was repeated with the measurement noise increased ten times to 1.0 mm. The entire process was again repeated using the low and high noise levels for the ballbar configuration.

In general, accuracy of parameter identification increased and the position error decreased as the number of observations increased regardless of noise level and calibration configuration used. For the low noise level, the calibrated manipulator accuracy is of the same order of magnitude as the attainable repeatability (0.15 mm) for this type of manipulator. This suggests that in the presence of a readily

attained low level of measurement noise (0.1 mm) the manipulator accuracy attained using the ballbar method is roughly equal to that attained using the unconstrained method and is, in fact, quite satisfactory. In addition, the manipulator accuracy can be improved by increasing the number of observations taken during the measurement phase of the calibration. There reaches a point, however, when making additional observations produces no marked improvement in manipulator accuracy. Table III is a table of the nominal and identified kinematic parameters for the unconstrained calibration using low noise and 60 observations. Table IV shows the same parameters for the ballbar calibration using low noise and 55 observations.

When the high noise level (1.0 mm) was used, the position error obtained using the ballbar method was unsatisfactory even when a large number of observations were made. For the unconstrained case, the position error was an order of magnitude higher than it was using the low noise level. The ability to obtain high manipulator accuracy is directly dependent on the effectiveness with which noise can be eliminated from the measurement process. In actual calibrations, the reduction of measurement noise is a pivotal step in the process.

For calibrations of serial link manipulators, the ballbar method has been proven to be a convenient alternative to the use of expensive ancillary measurement equipment. With this in

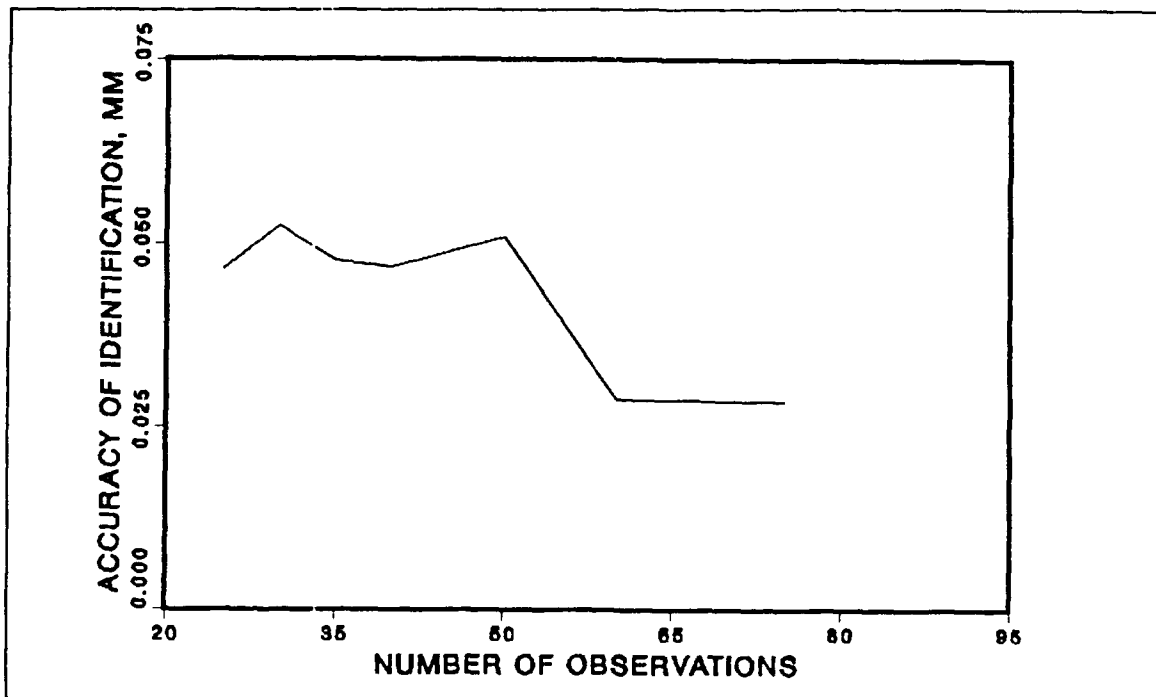


Figure 11. Accuracy of Parameter Identification/Low Noise (Unconstrained Method).

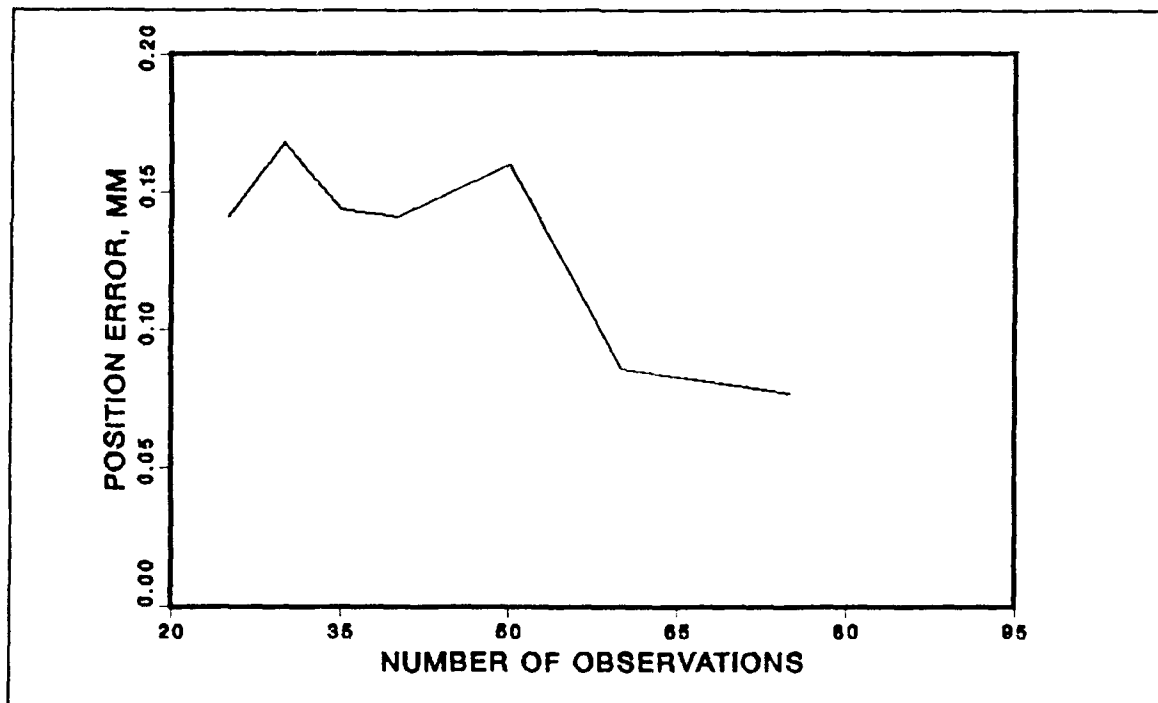


Figure 12. Manipulator Accuracy/Low Noise (Unconstrained Method).

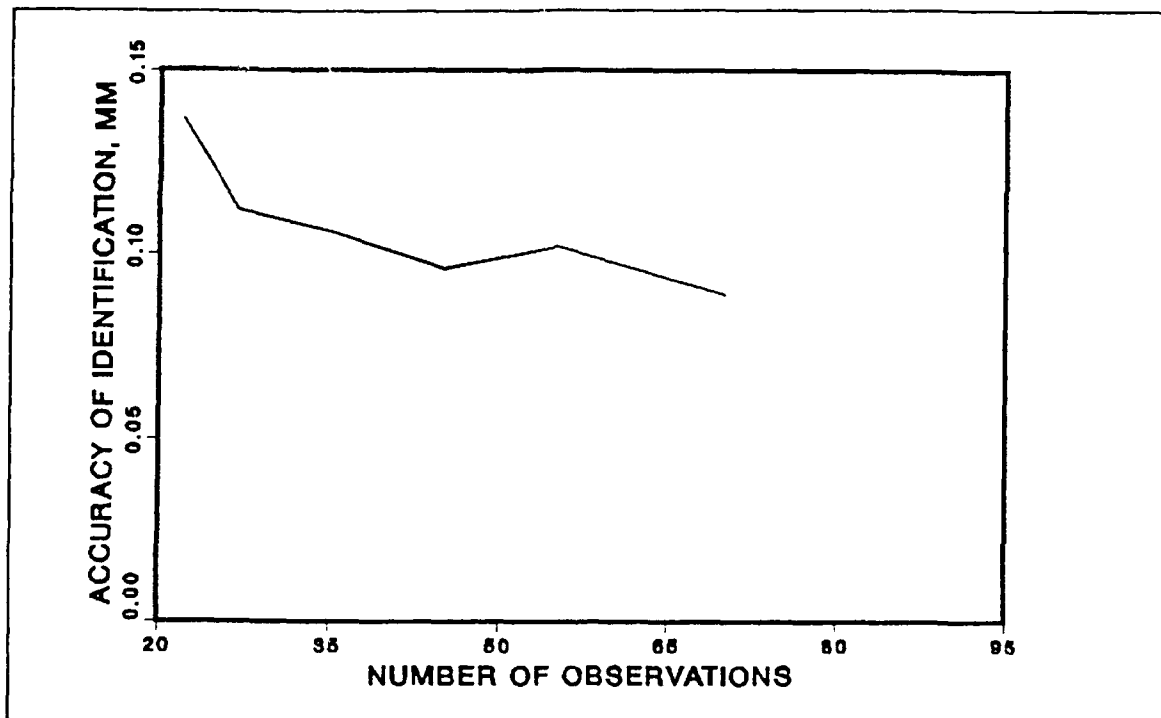


Figure 13. Accuracy of Parameter Identification/Low Noise (Ballbar Method).

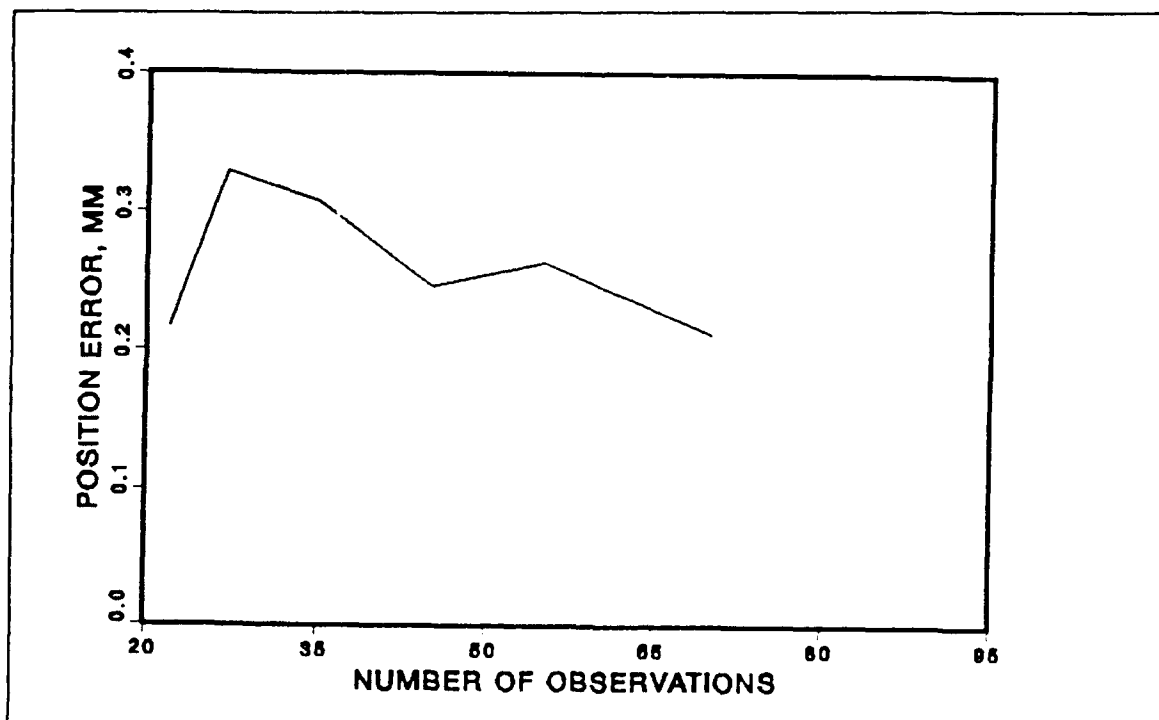


Figure 14. Manipulator Accuracy/Low Noise (Ballbar Method).

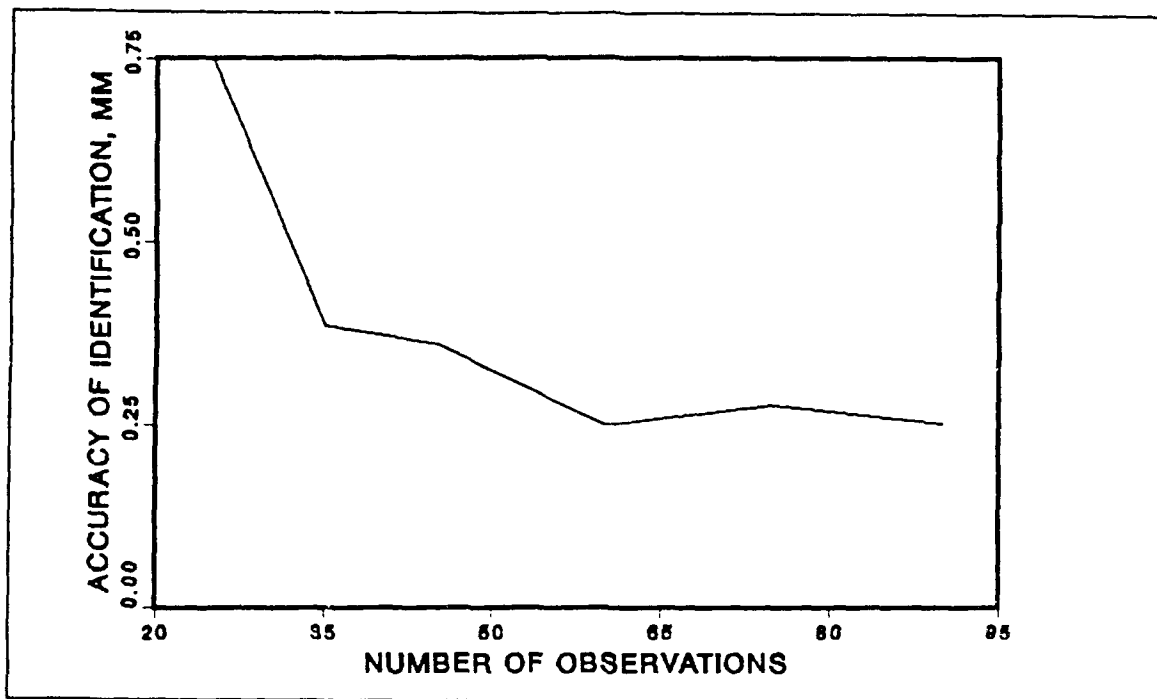


Figure 15. Accuracy of Parameter Identification/Noise x 10 (Unconstrained Method).

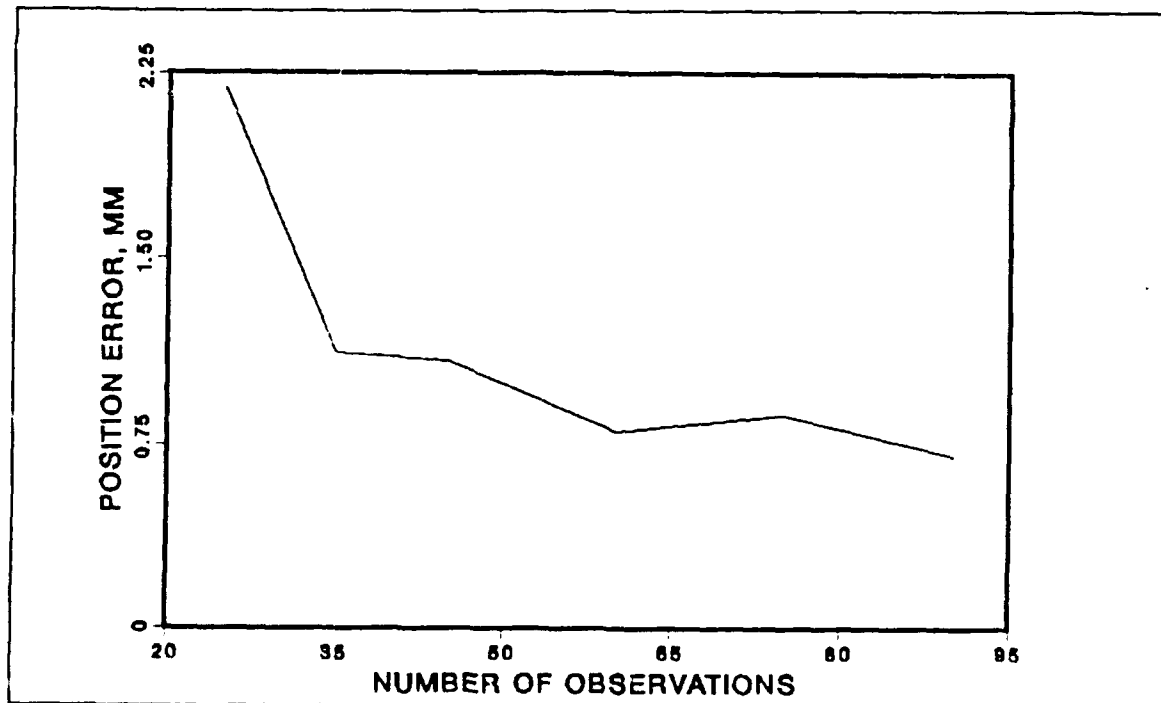


Figure 16. Manipulator Accuracy/Noise x 10 (Unconstrained Method).

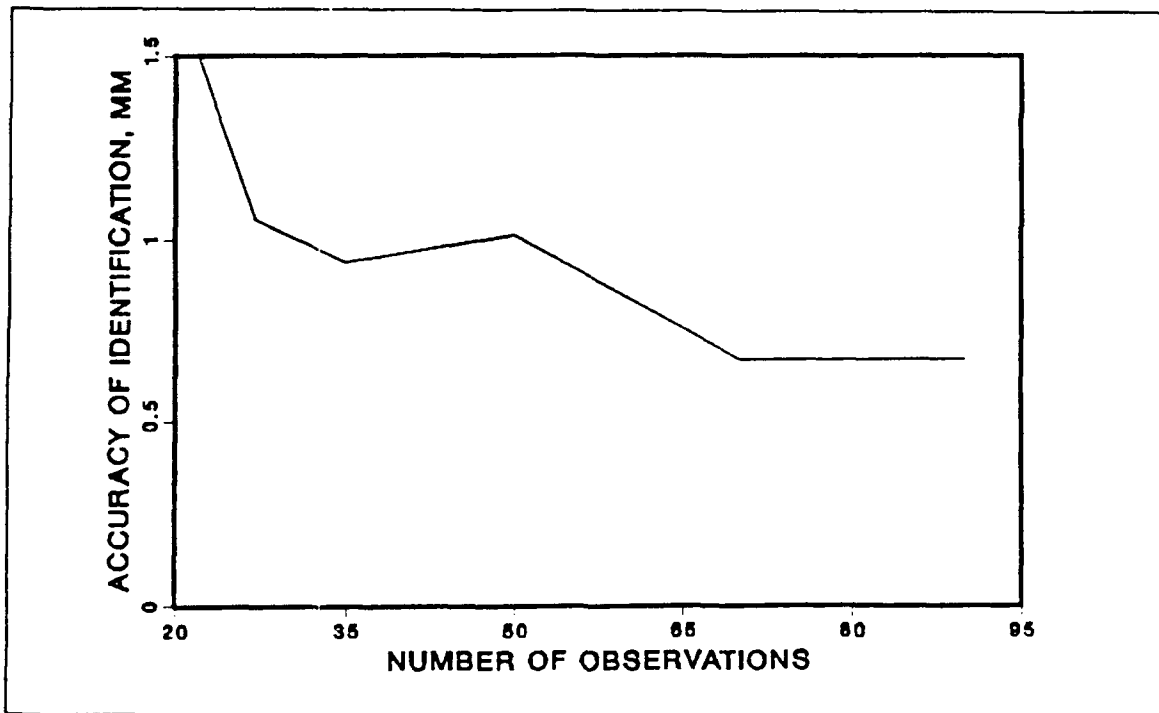


Figure 17. Accuracy of Parameter Identification/Noise x 10 (Ballbar Method).

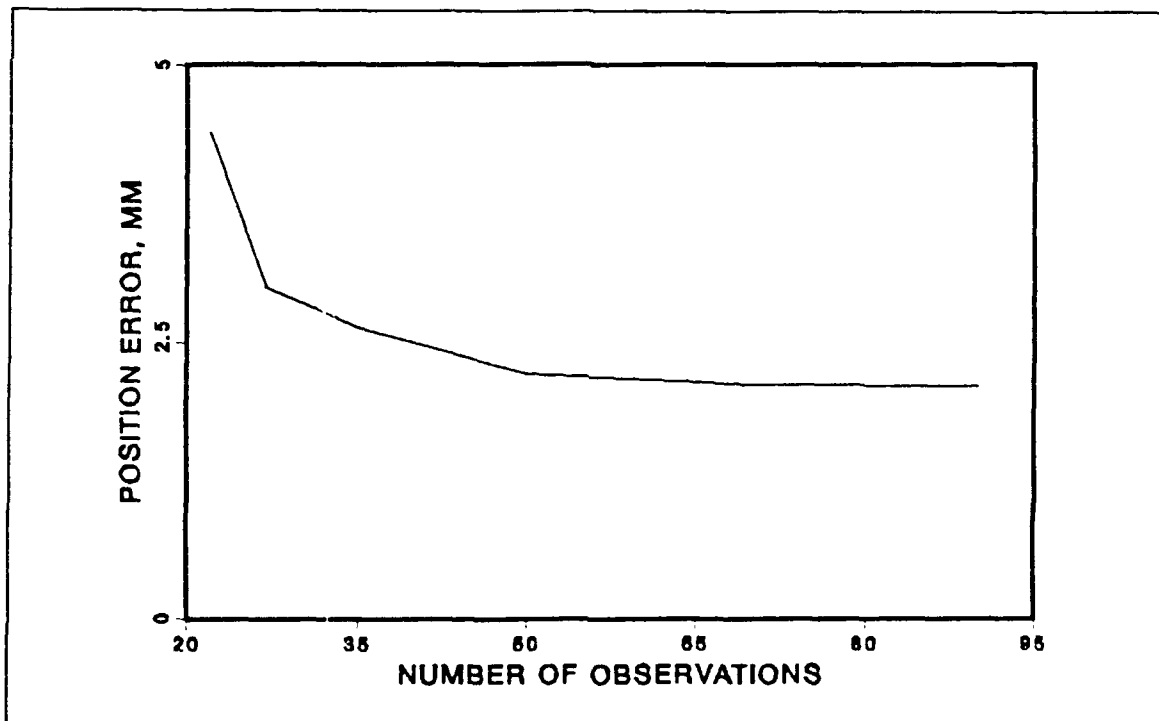


Figure 18. Manipulator Accuracy/Noise x 10 (Ballbar Method).

TABLE III. NOMINAL AND IDENTIFIED KINEMATIC PARAMETERS USING UNCONSTRAINED METHOD WITH LOW NOISE AND 60 OBSERVATIONS

PARAMETER	NOMINAL VALUE	IDENTIFIED VALUE
$\delta\theta_1$	0.0	1.00093
d_1	0.0	0.26324
a_1	1492.3	1492.54560
α_1	90.0	91.00015
$\delta\theta_2$	0.0	0.99907
d_2	0.0	0.28461
a_2	0.0	0.25271
α_2	-90.0	-89.00049
$\delta\theta_3$	90.0	90.99987
d_3	0.0	0.24319
a_3	0.0	0.25737
α_3	-90.0	-88.99892
d_4	730.3	730.55844
α_4	0.0	1.00002
β	0.0	0.99810
$\delta\theta_5$	-90.0	-89.00627
a_5	82.55	82.78718
α_5	90.0	90.97766
$\delta\theta_6$	90.0	91.02502
d_6	0.0	0.26702
a_6	0.0	0.23813
α_6	90.0	91.04527
$\delta\phi_7$	0.0	0.99364
Px_7	50.8	51.01315
Pz_7	50.8	51.04286

hand, an effort was made to determine which world coordinate frame locations yielded the smallest manipulator position error. Referring back to Figure 8, the x-coordinate of the world coordinate frame which is the distance to the first joint axis was set at each of the following three values:

TABLE IV. NOMINAL AND IDENTIFIED KINEMATIC PARAMETERS USING BALLBAR METHOD WITH LOW NOISE AND 55 OBSERVATIONS

PARAMETER	NOMINAL VALUE	IDENTIFIED VALUE
x_w	1542.7	1543.20468
y_w	-132.95	-132.73788
d_1	0.0	0.23486
a_1	0.0	0.26105
α_1	-90.0	-89.00081
$\delta\theta_2$	90.0	91.00244
d_2	0.0	0.31004
a_2	0.0	0.30212
α_2	-90.0	-89.00261
d_3	0.0	0.42076
α_3	0.0	1.00434
β_1	0.0	1.00388
$\delta\theta_4$	-90.0	-89.00484
a_4	82.55	82.81179
α_4	90.0	90.98414
$\delta\theta_5$	90.0	91.01540
d_5	0.0	0.24605
a_5	0.0	0.23570
α_5	90.0	90.98934
$\delta\phi_6$	0.0	1.01387
Px_6	50.8	50.65428
Pz_6	50.8	50.65121

$x=1292.7$ mm, 1542.7 mm, and 1792.7 mm. At each of these x -coordinate levels, the world coordinate frame was positioned at each node of a five by five y - z grid and the position error

was computed at each location. Figure 19, Figure 20, and Figure 21 illustrate the results. The location of the world coordinate frame does not significantly influence the position error obtained as long it is within the manipulators working volume.

The ballbar method is ideal for calibration of industrial robots in that it is quick, inexpensive, and simple to perform. The manipulator can be calibrated in place without the use of expensive measurement and the ballbar calibration method can produce manipulator accuracy of the same order of magnitude as other more costly and tedious methods.

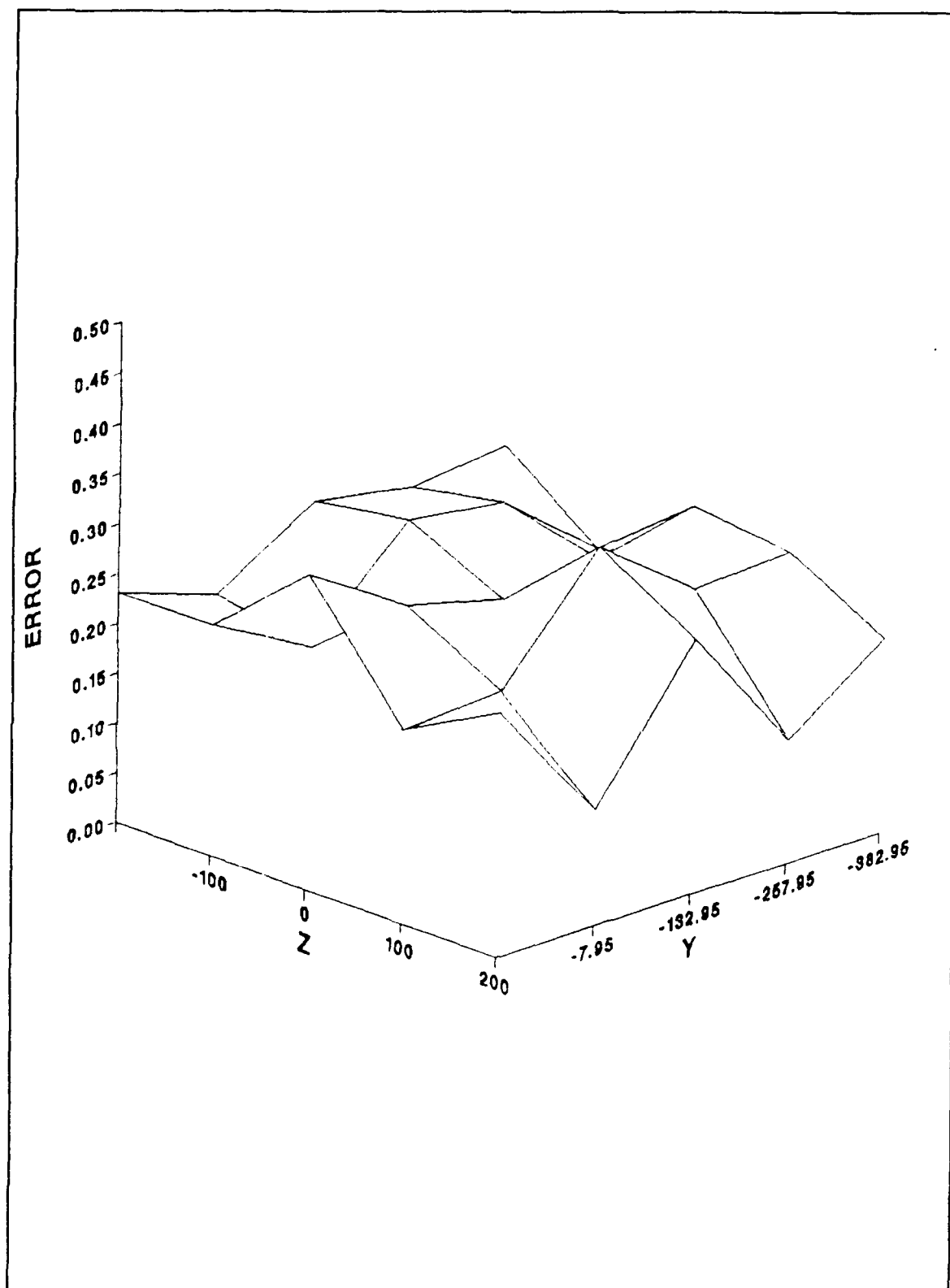


Figure 19. Position Error at $x=1292.7$ mm.

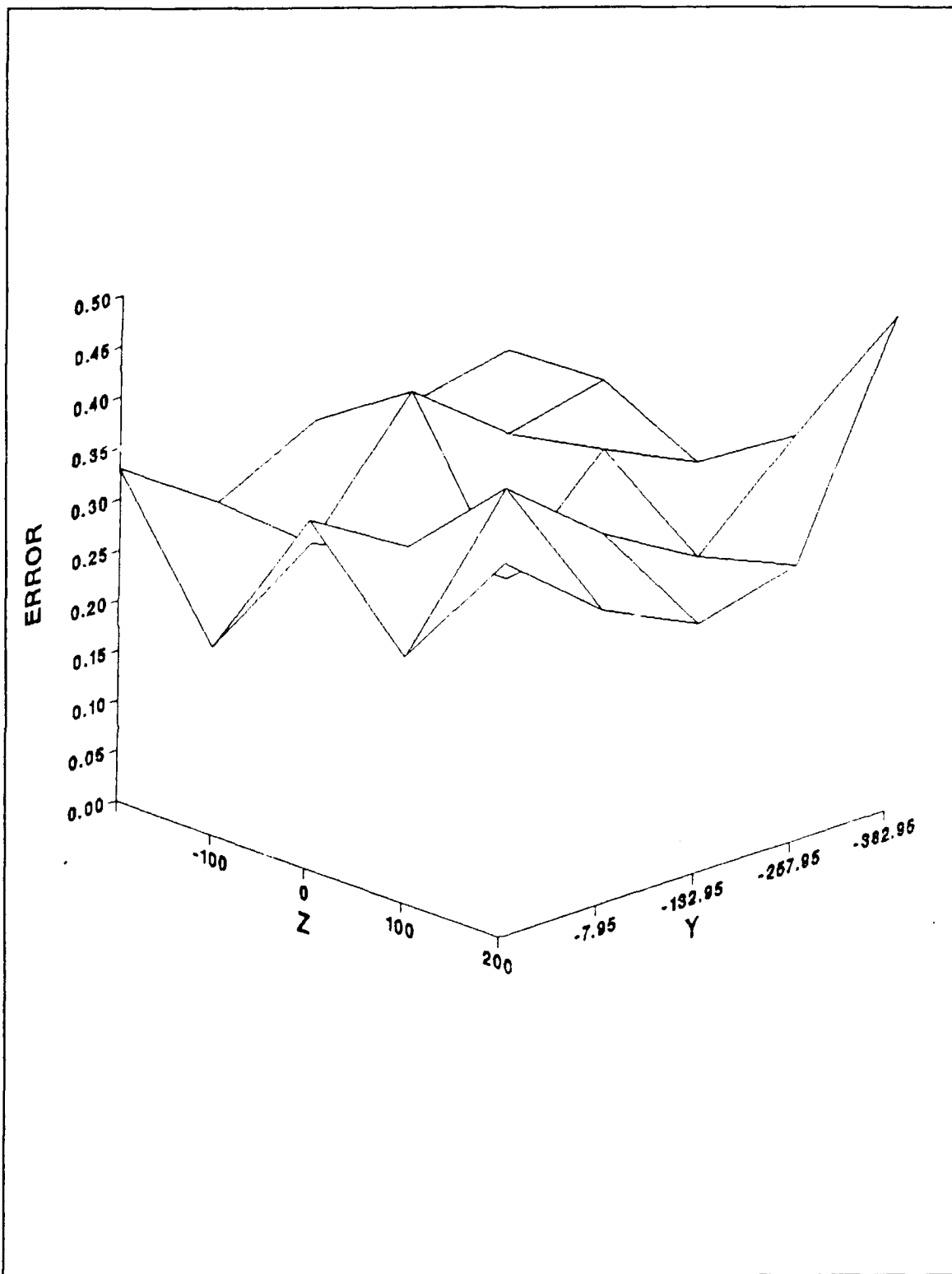


Figure 20. Position Error at $x=1542.7$ mm.

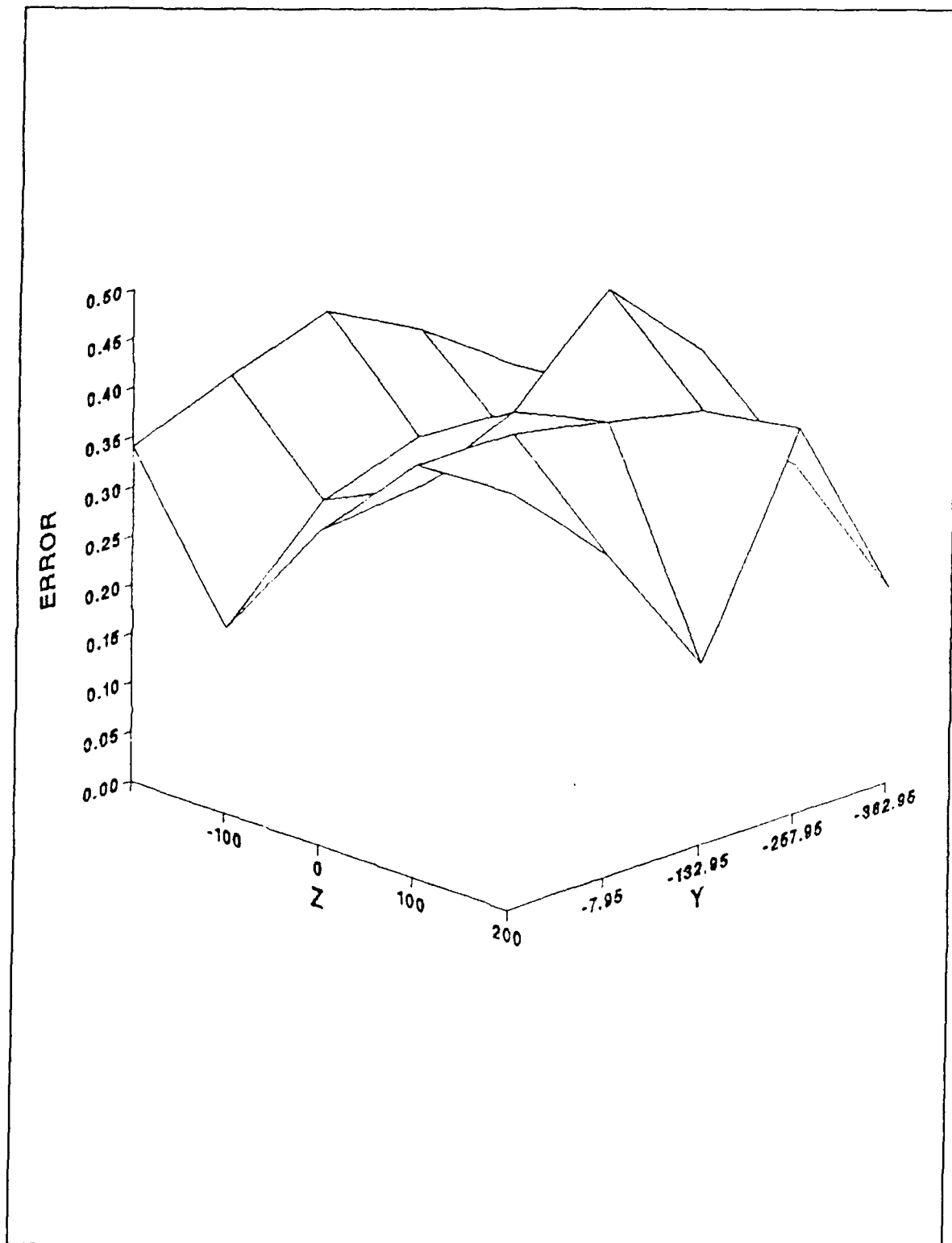


Figure 21. Position Error at $x=1792.7$ mm.

V. CONCLUSIONS

- In general, accuracy of parameter identification and manipulator accuracy increased as the number of observations taken increased.
- Accuracy of calibration process is directly related to the extent that measurement noise is reduced.
- Ballbar method produces manipulator accuracy of the same order as the unconstrained method and it is less expensive, quicker, and easier to perform.
- The location of the ballbar does not significantly influence the accuracy of the calibration as long as it is within the robot's working volume.

APPENDIX A: ZXSSQ

ZXSSQ is a Levenberg-Marquardt finite difference routine for solving non-linear least squares. The problem can be stated as follows:

Given M non-linear functions F_1, F_2, \dots, F_m of a vector parameter x , minimize over x

$$F_1(x)^2 = F_2(x)^2 + \dots + F_m(x)^2$$

where $x = (x_1, x_2, \dots, x_N)$ is a vector of N parameters to be estimated. When fitting a nonlinear model to data, the functions F_i should be defined as follows:

$$F_i(x) = y_i - g(x; v^i) \quad i = 1, 2, \dots, M$$

where y_i is the i^{th} observation of the dependent variable

$v^i = (v_1^i, v_2^i, \dots, v_{NV}^i)$ is a vector containing the i^{th} observation of the NV independent variables

g is the function defining the non-linear model

ZXSSQ is based on a modification of the Levenberg-Marquardt algorithm which eliminates the need for explicit derivatives. Let x^0 be an initial estimate of x . A sequence of approximations to the minimum point is generated by

$$x^{n+1} = x^n - [\alpha_n D_n + J_n^T J_n]^{-1} J_n^T F(x^n)$$

where J_n is the numerical Jacobian matrix evaluated at x^n

D_n is a diagonal matrix equal to the diagonal of $J_n^T J_n$

α_n is a positive scaling factor

When forward differences are used, the Jacobian is calculated by

$$\frac{1}{h_j} [F_i (x + h_j u_j) - F_i (x)]$$

where u_j is the j^{th} unit vector

$$h_j = \max(|x_j|, 0.1) \text{ eps}^{0.5}$$

eps is the relative precision of floating point arithmetic

For central difference, the Jacobian is as follows

$$\frac{1}{2h_j} [F_i (x + h_j u_j) - F_i (x - h_j u_j)]$$

Finally,

$$J_{n+1} = J_n + \frac{1}{\delta^T \delta} [F (x^{n+1}) - F (x^n) - J_n \delta] \delta^T$$

APPENDIX B: COMPUTER PROGRAMS

C*****

PROGRAM JOINT

C This program generates the joint variable data for the
C MODEL G manipulator simulation by random methods.

INTEGER I, J, K, NOBS, MAXNOBS
PARAMETER (MAXNOBS=360)
REAL Q(MAXNOBS,6), QMIN(6), QMAX(6)

COMMON /C1/ Q, QMAX, QMIN

C DATA QMIN/ -160.0, -223.0, -52.0, -110.0, -100.0, -266.0 /
C DATA QMAX/ 160.0, 43.0, 232.0, 170.0, 100.0, 266.0 /
C WRITE (6,*) 'Volume is MAX-POSSIBLE'

DATA QMIN/ -180.0, -180.0, 0.0, -180.0, -180.0, -180.0 /
DATA QMAX/ 180.0, 180.0, 762.0, 180.0, 180.0, 180.0 /
WRITE (6,*) 'Volume is FULL'

C DATA QMIN/ -90.0, -90.0, -90.0, -90.0, -90.0, -90.0 /
C DATA QMAX/ 90.0, 90.0, 90.0, 90.0, 90.0, 90.0 /
C WRITE (6,*) 'Volume is HALF'

C DATA QMIN/ -45.0, -45.0, -45.0, -45.0, -45.0, -45.0 /
C DATA QMAX/ 45.0, 45.0, 45.0, 45.0, 45.0, 45.0 /
C WRITE (6,*) 'Volume is QUARTER'

C Open output data file

OPEN (18, NAME='TELE-VAR.DAT', STATUS='NEW')

C Input number of observations from data file

OPEN (19, NAME='INPUT.DAT', STATUS='OLD')
DO I=1,10
READ (19,*)
ENDDO

READ (19,*) NOBS
WRITE(*,*) 'NOBS=',NOBS
CLOSE (19)

C Call the generation routine

CALL MSPREAD (NOBS)

C Save the joint variable data

DO II = 1, NOBS
WRITE (18,*) Q(II,1),Q(II,2),Q(II,3),Q(II,4),Q(II,5),Q(II,6)

```

        ENDDO

        CLOSE (18)
        STOP
        END

C *****

        SUBROUTINE MSPREAD (NOBS)

C This subroutine generates the joint data by the Monte Carlo method.
C The six joint variables are generated from six independant
C uniform random variables.

        INTEGER I, J, K, NOBS, MAXNOBS
        PARAMETER (MAXNOBS=360)
        REAL Q(MAXNOBS,6), QMIN(6), QMAX(6)
        INTEGER*4 ISEED
        REAL MAGQ(6), NUM

        COMMON /C1/ Q, QMAX, QMIN

C Get the random seed

        WRITE (6,*) 'Type in a 6-digit random number seed'
        READ (5,*) ISEED

C Calculate the Scaling factor for each random variable

        DO I = 1, 6
            MAGQ(I) = QMAX(I)-QMIN(I)
        ENDDO

C Generate the joint data

        DO J = 1, NOBS
            DO I = 1, 6

                CALL RANDOM (ISEED,NUM)

                Q(J,I) = QMIN(I) + MAGQ(I) * NUM
            ENDDO
        ENDDO

        RETURN
        END

C *****

        SUBROUTINE RANDOM (x,z)

        REAL FM, FX, Z
        INTEGER A, X, I, M
        DATA I/1/

        IF ( I .EQ. 0 ) GO TO 1000
        I=0
        M= 2 ** 20
        FM= M
        A= 2**10 + 3

```

```
1000  X= MOD ( A*X ,M)
      FX= X
      Z= FX/ FM

      RETURN
      END
```

```
C *****
```

C *****

PROGRAM POSE

C This program generates the pose data for the MODEL G manipulator
C simulation. It reads the joint variable data from file TELE-VAR.DAT.

```
INTEGER*4 ISEED
REAL*8 RNX, RNY, RNZ, MAGNX, MAGN1
REAL*8 RN1, RN2, RN3, RN4, RN5, RN6
INTEGER I, J, K, NOBS, MAXNOBS, N
PARAMETER (MAXNOBS=360)
REAL*8 DANGLE, DLENTN
REAL*8 PI
PARAMETER (PI=3.141592653589793)

REAL*8 DTW, DT1, DT2, DT3, DT4, DT5
REAL*8 DDW, DD1, DD2, DD3, DD4, DD5
REAL*8 AAW, AA1, AA2, AA3, AA4, AA5
REAL*8 ALW, AL1, AL2, AL3, AL4, AL5
REAL*8 BLW, BL1, BL2, BL3, BL4, BL5
REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6, D3
REAL*8 THETA1, THETA2, THETA3, THETA4, THETA5, THETA6
REAL*8 THW, TH1, TH2, TH3, TH4, TH5
REAL*8 TW(4,4), T1(4,4), T2(4,4), T3(4,4)
REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
REAL*8 TIMAT(4,4), T(4,4)
```

C Initialize the TIMAT matrix to an I matrix:

```
DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/
```

C Get the random number seed

```
WRITE (6,*) 'Type in a 6-digit random number seed'
READ (5,*) ISEED
```

C Open input files and output data file

```
OPEN (8, NAME='TELE-VAR.DAT', STATUS='OLD')
OPEN (9, NAME='TELE-POS.DAT', STATUS='NEW')
OPEN (10, NAME='INPUT.DAT', STATUS='OLD')
```

C Input parameters

```
read (10,*)
read (10,*) dtw,ddw,aaw,alw,blw
read (10,*) dt1,dd1,aa1,al1,bl1
read (10,*) dt2,dd2,aa2,al2,bl2
read (10,*) dt3,dd3,aa3,al3,bl3
read (10,*) dt4,dd4,aa4,al4,bl4
read (10,*) dt5,dd5,aa5,al5,bl5
read (10,*)
read (10,*) df6,th6,si6,px6,py6,pz6
read (10,*)
read (10,*) nob,n,dangle,dlentn,magnx,magn1
```

C Add encoder Offsets:

```
DTW = DTW + DANGLE
DT1 = DT1 + DANGLE
```

```

DT2 = DT2 + DANGLE
DT3 = DT3          ! defined
DT4 = DT4 + DANGLE
DT5 = DT5 + DANGLE

```

C Set link parameters for the manipulator:

```

ALW = ALW + DANGLE
AL1 = AL1 + DANGLE
AL2 = AL2 + DANGLE
AL3 = AL3 + DANGLE
AL4 = AL4 + DANGLE
AL5 = AL5 + DANGLE

AAW = AAW + DLENT
AA1 = AA1 + DLENT
AA2 = AA2 + DLENT
AA3 = AA3          ! defined
AA4 = AA4 + DLENT
AA5 = AA5 + DLENT

DDW = DDW + DLENT
DD1 = DD1 + DLENT
DD2 = DD2 + DLENT
DD3 = DD3 + DLENT
DD4 = DD4          ! defined
DD5 = DD5 + DLENT

BLW = BLW          ! defined
BL1 = BL1          ! defined
BL2 = BL2          ! defined
BL3 = BL3 + DANGLE
BL4 = BL4          ! defined
BL5 = BL5          ! defined

DF6 = DF6 + DANGLE
TH6 = 0.0
SI6 = 0.0
PX6 = PX6 + DLENT
PY6 = 0.0
PZ6 = PZ6 + DLENT
D3 = DD3

```

C Loop NOBS times

```

DO I = 1, NOBS

```

C Initialize the T matrix to an I matrix:

```

DO J=1,4
DO K=1,4
T(J,K) = TIMAT(J,K)
ENDDC
ENDDO

```

C Manipulator joint angle input:

```

READ (8,*) THETA1, THETA2, THETA3, THETA4, THETA5, THETA6

THW = DTW
TH1 = DT1 + THETA1

```

```

TH2 = DT2 + THETA2
TH3 = DT3
TH4 = DT4 + THETA4
TH5 = DT5 + THETA5
FI6 = DF6 + THETA6
DD3 = D3 + THETA3

```

C Compute the T matrices, TW thru T6:

```

CALL TRANSFORM ( ALW, AAW, DDW, THW, BLW, TW )
CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

CALL T3RPY ( FI6, TH6, SI6, TRPY )
CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
CALL MATMULC ( T6, TRPY, TXYZ )

```

C Compute the overall transformation, T:

```

CALL MATMULA ( T, TW )
CALL MATMULA ( T, T1 )
CALL MATMULA ( T, T2 )
CALL MATMULA ( T, T3 )
CALL MATMULA ( T, T4 )
CALL MATMULA ( T, T5 )
CALL MATMULA ( T, T6 )

```

C Generate the random noise

```

CALL RANDOM(ISEED,RNX)
CALL RANDOM(ISEED,RNY)
CALL RANDOM(ISEED,RNZ)

CALL RANDOM(ISEED,RN1)
CALL RANDOM(ISEED,RN2)
CALL RANDOM(ISEED,RN3)
CALL RANDOM(ISEED,RN4)
CALL RANDOM(ISEED,RN5)
CALL RANDOM(ISEED,RN6)

RNX = MAGNX*( 2.0*RNX - 1.0 )
RNY = MAGNX*( 2.0*RNY - 1.0 )
RNZ = MAGNX*( 2.0*RNZ - 1.0 )

RN1 = MAGN1*( 2.0*RN1 - 1.0 )
RN2 = MAGN1*( 2.0*RN2 - 1.0 )
RN3 = MAGN1*( 2.0*RN3 - 1.0 )
RN4 = MAGN1*( 2.0*RN4 - 1.0 )
RN5 = MAGN1*( 2.0*RN5 - 1.0 )
RN6 = MAGN1*( 2.0*RN6 - 1.0 )

```

C Add noise to measurements and encoder readings

```

T(1,4) = T(1,4) + RNX
T(2,4) = T(2,4) + RNY
T(3,4) = T(3,4) + RNZ

THETA1 = THETA1 + RN1

```

```

THETA2 = THETA2 +RN2
THETA3 = THETA3 +RN3
THETA4 = THETA4 +RN4
THETA5 = THETA5 +RN5
THETA6 = THETA6 +RN6

```

C Store the manipulator joint vector and measured tool pose

```

WRITE (9,991) THETA1, THETA2, THETA3, THETA4, THETA5, THETA6
WRITE (9,992) T(1,4)
WRITE (9,992) T(2,4)
WRITE (9,992) T(3,4)
WRITE (9,*)

```

C Format below decides the digits of accuracy of simulation data

```

991 FORMAT ( 6F12.6 )!Joint vector data
992 FORMAT ( F12.5 )      !Measurement data

```

C End do-loop for counter I

```

ENDDO

WRITE (6,*) 'Data stored in F12.5, F12.4 format'

CLOSE (8)
CLOSE (9)
STOP
END

```

C *****

SUBROUTINE RANDOM (X,Z)

```

REAL FM, FX, Z
INTEGER A, X, I, M
DATA I/1/

```

```

IF ( I .EQ. 0 ) GO TO 1000
I=0
M= 2 ** 20
FM= M
A= 2**10 + 3

```

```

1000      X= MOD( A*X ,M)
FX= X
Z= FX/ FM

```

```

RETURN
END

```

C *****

C *****

PROGRAM TELEBAR

C This program generates a set of joint angles for the calibration
C of the MODEL G manipulator using a ball bar to constrain the end
C point of the manipulator.

```
INTEGER LDFJAC, M, N, obs, nobs
PARAMETER (LDFJAC=3, M=LDFJAC, N=6)

REAL*8 DT1, DT2, DT3, DT4, DT5
REAL*8 DD1, DD2, DD3, DD4, DD5
REAL*8 AA1, AA2, AA3, AA4, AA5
REAL*8 AL1, AL2, AL3, AL4, AL5
REAL*8 BL1, BL2, BL3, BL4, BL5
REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
REAL*8 XW, YW, ZW
```

```
INTEGER infer, ier, iopt, nsig, maxfn
REAL*8 FJAC(LDFJAC, N), xjtj((n+1)*n/2), xjac(ldfjac, n)
REAL*8 parm(4), f(ldfjac), work((5*n)+(2*m)+((n+1)*n/2))
REAL*8 X(N)
real*8 r, phimax, phimin, thetamax, thetamin, phi, theta
real*8 xb, yb, zb, ssq, rr, magnx, magnl
```

EXTERNAL TELE_ARM

```
INTEGER I, J, K
REAL*8 TDES(4, 4), qmax(6), qmin(6), SCALE, DANGLE, DLENTN, NUM
COMMON /PDATA/ TDES, DANGLE, DLENTN, r
COMMON /KIN/ DT1, DT2, DT3, DT4, DT5,
&          AL1, AL2, AL3, AL4, AL5,
&          AA1, AA2, AA3, AA4, AA5,
&          DD1, DD2, DD3, DD4, DD5,
&          BL1, BL2, BL3, BL4, BL5,
&          XW, YW, ZW,
&          DF6, TH6, SI6, PX6, PY6, PZ6
```

C Joint angle ranges

```
data qmin/-30.0, -45.0, 0.0, -180.0, 0.0, -180.0/
data qmax/25.0, 45.0, 762.0, 180.0, 90.0, 180.0/
```

C Initialize data variables

```
obs=0
```

C Open data files for input

```
OPEN (10, NAME='TELE-SOLN.DAT', STATUS='NEW')
open (9, NAME='INPUT.DAT', STATUS='old')
```

C Read input kinematic data

```
read (9, *)
read (9, *) xw, yw, zw
read (9, *) dt1, dd1, aa1, al1, bl1
read (9, *) dt2, dd2, aa2, al2, bl2
read (9, *) dt3, dd3, aa3, al3, bl3
read (9, *) dt4, dd4, aa4, al4, bl4
```



```

      read (9,*) dt5,dd5,aa5,a15,b15
      read (9,*)
      read (9,*) df6,th6,si6,px6,py6,pz6
      read (9,*)
      read (9,*) nobs,r,dangle,dlenth,magnx,magnl

```

```

      close (9)

```

C Adjust nominal values

```

      xw=xw+dlenth
      yw=yw+dlenth

      dt2=dt2+dangle
      dt4=dt4+dangle
      dt5=dt5+dangle

      a11=a11+dangle
      a12=a12+dangle
      a13=a13+dangle
      a14=a14+dangle
      a15=a15+dangle

      aa1=aa1+dlenth
      aa2=aa2+dlenth
      aa4=aa4+dlenth
      aa5=aa5+dlenth

      dd1=dd1+dlenth
      dd2=dd2+dlenth
      dd3=dd3+dlenth
      dd5=dd5+dlenth

      b13=b13+dangle

      df6=df6+dangle
      px6=px6+dlenth
      pz6=pz6+dlenth

```

C Limits on bar rotation

```

      phimax=180.0
      phimin=-180.0
      thetamax=90.0
      thetamin=-90.0

```

C Get random number seed

```

c      ISEED = 123456

      write (6,*) 'Type in a 6-digit random number seed'
      read (5,*) iseed

```

C Write NOBS to TELE-SOLN.DAT

```

      write (10,*) nobs

```

C Start of main loop

```

1010      obs=obs+1

```

C Set joint angles to zero

```
do i=1,n
x(i)=0.0
enddo
```

C Get random bar angles

```
1000    call random (iseed,num)
phi=phimin+(phimax-phimin)*num
call random (iseed,num)
theta=thetamin+(thetamax-thetamin)*num
```

C Calculate end point of the bar

```
xb=r*cosd(theta)
yb=r*sind(theta)*cosd(phi)
zb=r*sind(theta)*sind(phi)
```

C Reacheability calculation

```
if (z .lt. 0.0) go to 1000
```

C Establish desired tool pose

```
do ii=1,4
do jj=1,4
TDES(ii,jj)=0.0
enddo
enddo
```

```
TDES(1,4)=xb
TDES(2,4)=yb
TDES(3,4)=zb
TDES(4,4) = 1.0
```

C Call IMSL ZXSSQ for inverse kinematic solution

```
nsig=4
eps=0.0
delta=0.0
maxfn=500
iopt=1
ixjac=ldfjac

CALL ZXSSQ(tele_arm,m,n,nsig,eps,delta,maxfn,iopt,parm,x,
&          ssq,f,xjac,ixjac,xjtj,work,infer,ier)
```

C Print results to 2 decimal places

```
write(6,*) obs,ssq,iseed
WRITE (10,*) X(1), X(2), X(3), X(4), X(5), X(6)
```

C Continue for other bar angles

```
if (obs .lt. nob) go to 1010

CLOSE (10)
```

```

        WRITE(6,*) XW,YW,ZW
    END

C *****

    SUBROUTINE tele_ARM (X,M,N,F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine ZXSSQ. It is the forward kinematic solution for
C the MODEL G manipulator.

    INTEGER M, N
    REAL*8 X(N), F(M)

    INTEGER II, JJ
    REAL*8 DT1, DT2, DT3, DT4, DT5
    REAL*8 DD1, DD2, DD3, DD4, DD5
    REAL*8 AA1, AA2, AA3, AA4, AA5
    REAL*8 AL1, AL2, AL3, AL4, AL5
    REAL*8 BL1, BL2, BL3, BL4, BL5
    REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
    REAL*8 XW, YW, ZW, D3

    REAL*8 TH1, TH2, TH3, TH4, TH5
    REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
    REAL*8 T5(4,4), T6(4,4), trpy(4,4), txyz(4,4)
    REAL*8 TIMAT(4,4), T(4,4)
    REAL*8 disq,dis

    INTEGER I, J, K
    REAL*8 TDES(4,4), DANGLE, DLENTN, r

    COMMON /PDATA/ TDES, DANGLE, DLENTN, r
    COMMON /KIN/ DT1,DT2,DT3,DT4,DT5,
&               AL1,AL2,AL3,AL4,AL5,
&               AA1,AA2,AA3,AA4,AA5,
&               DD1,DD2,DD3,DD4,DD5,
&               BL1,BL2,BL3,BL4,BL5,
&               XW,YW,ZW,
&               DF6,TH6,SI6,PX6,PY6,PZ6

C Initialize the TIMAT matrix to an I matrix:

    DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Initialize the T matrix to an I matrix

    DO II = 1,4
    DO JJ = 1,4
        T(II,JJ) = TIMAT(II,JJ)
    ENDDO
    ENDDO

C Manipulator joint angles

    TH1 = DT1 + X(1)
    TH2 = DT2 + X(2)
    TH3 = DT3
    TH4 = DT4 + X(4)
    TH5 = DT5 + X(5)
    FI6 = DF6 + X(6)

```

D3 = DD3 + X(3)

C Compute the T matrices, T1 thru T6:

CALL t3xyz (xw,yw,zw,T0)

CALL TRANSFORM (AL1, AA1, DD1, TH1, BL1, T1)

CALL TRANSFORM (AL2, AA2, DD2, TH2, BL2, T2)

CALL TRANSFORM (AL3, AA3, D3, TH3, BL3, T3)

CALL TRANSFORM (AL4, AA4, DD4, TH4, BL4, T4)

CALL TRANSFORM (AL5, AA5, DD5, TH5, BL5, T5)

CALL t3rpy (fi6, th6, si6, trpy)

CALL T3XYZ (PX6, P16, PZ6, txyz)

CALL matmulc (t6, trpy, txyz)

C Compute the overall transformation, T:

CALL MATMULA (T, T0)

CALL MATMULA (T, T1)

CALL MATMULA (T, T2)

CALL MATMULA (T, T3)

CALL MATMULA (T, T4)

CALL MATMULA (T, T5)

CALL MATMULA (T, T6)

C Calculate the function F

f(1)=t(1,4)-tdes(1,4)

f(2)=t(2,4)-tdes(2,4)

f(3)=t(3,4)-tdes(3,4)

RETURN

END

C *****

SUBROUTINE RANDOM (x,z)

C This subroutine generates random numbers in the range 0-1

C using a supplied seed x, the returned random number being z.

REAL FM, FX, Z
INTEGER A, X, I, M
DATA I/1/

IF (I .EQ. 0) GO TO 1000

I=0

M= 2 ** 20

FM= M

A= 2**10 + 3

1000 X= MOD(A*X ,M)

FX= X

Z= FX/ FM

RETURN

END

C *****

C *****

PROGRAM ID6

C Robot Identification using the Non-linear Least Squares method.
C Simulation data is read for the MODEL G manipulator from
C the data file TELE-SOLN.DAT

C Change parameter LDFJAC to change the number of observations,
C set LDFJAC = Number of observations

INTEGER LDFJAC, MM, M, NN, N, NSIG, MAXFN, IOPT, IXJAC, INFER, IER
PARAMETER (LDFJAC=90, MM=LDFJAC, NN=22)

REAL*8 FJAC(LDFJAC,NN), XJTJ((NN+1)*NN/2)
REAL*8 PARM(4), F(LDFJAC), WORK((5*NN)+(2*MM)+((NN+1)*NN/2))
REAL*8 X(NN)
EXTERNAL TELE_ARM

REAL*8 DANGLE, DLENT, TQ, DQ, EPS, DELTA, SSQ
REAL*8 SQERR1, SQERR2
REAL*8 XW,YW,ZW
REAL*8 DT1, DT2, DT3, DT4, DT5
REAL*8 DD1, DD2, DD3, DD4, DD5
REAL*8 AA1, AA2, AA3, AA4, AA5
REAL*8 AL1, AL2, AL3, AL4, AL5
REAL*8 BL1, BL2, BL3, BL4, BL5
REAL*8 FI6, DF6, TH6, SI6, PX6, PY6, PZ6

INTEGER I, J, K, NOBS, MAXNOBS
REAL*8 magnx,magnl
PARAMETER (MAXNOBS=100)
REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
REAL*8 R
COMMON /PDATA/ NOBS, TET1, TET2, TET3, TET4, TET5, TET6, R

COMMON /KIN/ DT1,DT2,DT3,DT4,DT5,
& AL1,AL2,AL3,AL4,AL5,
& AA1,AA2,AA3,AA4,AA5,
& DD1,DD2,DD3,DD4,DD5,
& BL1,BL2,BL3,BL4,BL5,
& XW,YW,ZW,
& DF6,TH6,SI6,PX6,PY6,PZ6

C Open data files for inputs and results

OPEN (8, NAME='RESULT.DAT', STATUS='NEW')
OPEN (9, NAME='TELE-SOLN.DAT', STATUS='OLD')
OPEN (10,NAME='INPUT.DAT', STATUS='OLD')

C Read input parameters

read (10,*)
read (10,*) xw,yw,zw
read (10,*) dt1,dd1,aa1,al1,bl1
read (10,*) dt2,dd2,aa2,al2,bl2
read (10,*) dt3,dd3,aa3,al3,bl3
read (10,*) dt4,dd4,aa4,al4,bl4
read (10,*) dt5,dd5,aa5,al5,bl5
read (10,*)

```

        read (10,*) df6,th6,si6,px6,py6,pz6
        read (10,*)
        read (10,*) nobs,r,dangle,dlenth,magnx,magnl

CLOSE (10)

C Initialize data variables

X(1)=XW
X(2)=YW

X(3)=DD1
    X(4)=AA1
X(5)=AL1

X(6)=DT2
X(7)=DD2
X(8)=AA2
X(9)=AL2

X(10)=DD3
X(11)=AL3
X(12)=BL3

X(13)=DT4
X(14)=AA4
X(15)=AL4

X(16)=DT5
X(17)=DD5
X(18)=AA5
X(19)=AL5

X(20)=DF6
X(21)=PX6
X(22)=PZ6

R=R+MAGNX

C Read simulated joint data and tool pose

READ (9,*) NOBS

DO J = 1, NOBS
    READ (9,*) TET1(J), TET2(J), TET3(J), TET4(J), TET5(J), TET6(J)
ENDDO
CLOSE (9)

C Call IMSL routine for non-linear identification

NSIG=4
EPS=0.0
DELTA=0.0
MAXFN=1500
IOPT=1
IXJAC=LDFJAC
M=NOBS

CALL ZXSSQ(TELE_ARM,M,NN,NSIG,EPS,DELTA,MAXFN,IOPT,
&          PARM,X,SSQ,F,FJAC,IXJAC,XJTJ,WORK,INFER,IER)

```

C Save results to data file

```
WRITE (8,*)  
WRITE (8,*) 'XW, YW, ZW'  
WRITE (8,*) X(1), X(2), ZW  
WRITE (8,*)  
WRITE (8,*) 'DT1, DD1, AA1, AL1, BL1'  
WRITE (8,*) 0.0, X(3), X(4), X(5), 0.0  
WRITE (8,*)  
WRITE (8,*) 'DT2, DD2, AA2, AL2, BL2'  
WRITE (8,*) X(6), X(7), X(8), X(9), 0.0  
WRITE (8,*)  
WRITE (8,*) 'DT3, DD3, AA3, AL3, BL3'  
WRITE (8,*) 0.0, X(10), 0.0, X(11), X(12)  
WRITE (8,*)  
WRITE (8,*) 'DT4, DD4, AA4, AL4, BL4'  
WRITE (8,*) X(13), 0.0, X(14), X(15), 0.0  
WRITE (8,*)  
WRITE (8,*) 'DT5, DD5, AA5, AL5, BL5'  
WRITE (8,*) X(16), X(17), X(18), X(19), 0.0  
WRITE (8,*)  
WRITE (8,*) ' DF6, TH6, SI6, PX6, PY6, PZ6, R'  
WRITE (8,*) X(20), 0.0, 0.0, X(21), 0.0, X(22), R
```

C Restore initial values of input parameters

```
open (10,name='input.dat',status='old')  
  
read (10,*)  
read (10,*) xw,yw,zw  
read (10,*) dt1,dd1,aa1,al1,bl1  
read (10,*) dt2,dd2,aa2,al2,bl2  
read (10,*) dt3,dd3,aa3,al3,bl3  
read (10,*) dt4,dd4,aa4,al4,bl4  
read (10,*) dt5,dd5,aa5,al5,bl5  
read (10,*)  
read (10,*) df6,th6,si6,px6,py6,pz6  
read (10,*)  
read (10,*) nob, r, dangle, dlenth, magnx, magnl  
  
CLOSE (10)
```

C Calculate root mean square error in identification

```
TQ = DANGLE  
DQ = DLENTH
```

C Error in identification (angular parameters)

```
SQERR1 =  
& (AL1+TQ-X(5))**2 + (DT2+TQ-X(6))**2 + (AL2+TQ-X(9))**2  
& + (AL3+TQ-X(11))**2  
& + (BL3+TQ-X(12))**2 + (DT4+TQ-X(13))**2  
& + (AL4+TQ-X(15))**2 + (DT5+TQ-X(16))**2  
& + (AL5+TQ-X(19))**2  
& + (DF6+TQ-X(20))**2  
SQERR1 = DSQRT( SQERR1/10 )
```

C Error in identification (length parameters)

```
SQERR2 =
```

```

& (DD1+DQ-X(3))**2 + (AA1+DQ-X(4))**2
& + (DD2+DQ-X(7))**2 + (AA2+DQ-X(8))**2
& + (DD3+DQ-X(10))**2 + (AA4+DQ-X(14))**2
& + (DD5+DQ-X(17))**2 + (AA5+DQ-X(18))**2
& + (PX6+DQ-X(21))**2 + (PZ6+DQ-X(22))**2
& + (xw+dq-x(1))**2 + (yw+dq-x(2))**2
SQERR2 = DSQRT( SQERR2/12 )

```

```

WRITE (8,*)
WRITE (8,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
WRITE (8,*) SQERR2, SQERR1
WRITE (6,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
WRITE (6,*) SQERR2, SQERR1

```

```

WRITE (8,*)
WRITE (8,*) 'INFER, IER, NOBS, NSIG'
WRITE (8,*) INFER, IER, NOBS, NSIG
WRITE (6,*) 'INFER, IER, NOBS, NSIG'
WRITE (6,*) INFER, IER, NOBS, NSIG
WRITE (8,*)

```

```

CLOSE (8)

```

```

END

```

```

C *****

```

```

SUBROUTINE TELE_ARM (X, M, N, F)

```

```

C This subroutine calculates the non-linear function for the use of
C the IMSL routine ZXSSQ. It is the forward kinematic solution for
C the MODEL G manipulator.

```

```

INTEGER M, N
REAL*8 X(N), F(M)

```

```

INTEGER II, JJ

```

```

REAL*8 XW, YW, ZW
REAL*8 DT1, DT2, DT3, DT4, DT5
REAL*8 DD1, DD2, DD3, DD4, DD5
REAL*8 AA1, AA2, AA3, AA4, AA5
REAL*8 AL1, AL2, AL3, AL4, AL5
REAL*8 BL1, BL2, BL3, BL4, BL5
REAL*8 fi6, df6, th6, si6, PX6, PY6, PZ6, D3

```

```

REAL*8 TH1, TH2, TH3, TH4, TH5
REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
REAL*8 T5(4,4), T6(4,4), trpy(4,4), txyz(4,4)
REAL*8 TIMAT(4,4), T(4,4)

```

```

INTEGER I, J, K, NOBS, MAXNOBS
PARAMETER (MAXNOBS=100)
REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
REAL*8 R, RR
COMMON /PDATA/ NOBS, TET1, TET2, TET3, TET4, TET5, TET6, R

```

```

COMMON /KIN/ DT1, DT2, DT3, DT4, DT5,
& AL1, AL2, AL3, AL4, AL5,

```



```

&          AA1,AA2,AA3,AA4,AA5,
&          DD1,DD2,DD3,DD4,DD5,
&          BL1,BL2,BL3,BL4,BL5,
&          XW,YW,ZW,
&          DF6,TH6,SI6,PX6,PY6,PZ6

```

C Initialize the TIMAT matrix to an I matrix:

```
DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/
```

C Set parameters for the manipulator:

```

XW = X(1)
YW = X(2)

DD1 = X(3)
AA1 = X(4)
AL1 = X(5)

DT2 = X(6)
DD2 = X(7)
AA2 = X(8)
AL2 = X(9)

DD3 = X(10)
AL3 = X(11)
BL3 = X(12)

DT4 = X(13)
AA4 = X(14)
AL4 = X(15)

DT5 = X(16)
DD5 = X(17)
AA5 = X(18)
AL5 = X(19)

DF6 = X(20)
PX6 = X(21)
PZ6 = X(22)

```

C Loop NOBS times

```

K = 0
DO J = 1, NOBS

```

C Initialize the T matrix to an I matrix

```

DO II = 1,4
DO JJ = 1,4
  T(II,JJ) = TIMAT(II,JJ)
ENDDO
ENDDO

```

C Manipulator joint angles

```

TH1 = DT1 + TET1(J)
TH2 = DT2 + TET2(J)
TH3 = DT3
TH4 = DT4 + TET4(J)
TH5 = DT5 + TET5(J)

```

```

      FI6 = DF6 + TET6(J)
      D3 = DD3 + TET3(J)

C Compute the T matrices, T1 thru T6:

      CALL T3XYZ (XW,YW,ZW,T0)

      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, D3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

      CALL t3rpy ( fi6, th6, si6, trpy )
      CALL T3XYZ ( PX6, PY6, PZ6, txyz )
      CALL matmulc ( t6, trpy, txyz )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

C Calculate the function F

      rr=dsqrt( t(1,4)*t(1,4)+t(2,4)*t(2,4)+t(3,4)*t(3,4) )
      f(j)=dabs( rr-r)

C End the do-loop for counter J

      ENDDO

C Compute RMS error

      sumsq=0.0
      do j=1, nobs
      sumsq=sumsq+f(j)*f(j)
      enddo
      rms=sqrt(sumsq/nobs)
      write (6,*) rms

      RETURN
      END

C *****

```

```

C *****

PROGRAM VERIFY

C This program generates the six-dof pose error for the MODEL G
manipulator.
C It contains the identified calibration parameters and the exact
parameter.
C It uses a data file of verification joint angle sets POSEVER.DAT, and
the
C file RESULT.DAT from the program ID6.

INTEGER I, J, K, NPOSES, N
REAL*8 DANGLE, DLENTH
REAL*8 P(200),OR(200),W1(200),W2(200),W3(200)
REAL*8 DT(5),dd(5),aa(5),al(5),bl(5), world(3)
REAL*8 EDT(5),EDD(5),EAA(5),EAL(5),EBL(5), eworld(3)
REAL*8 EDF6, EFI6, ETH6, ESI6, EPX6, EPY6, EPZ6
REAL*8 THETA(1000,6), TDELTA(4,4)
REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
REAL*8 TIMAT(4,4), T(4,4), et(4,4)

REAL*8 DT1, DT2, DT3, DT4, DT5
REAL*8 DD1, DD2, DD3, DD4, DD5
REAL*8 AA1, AA2, AA3, AA4, AA5
REAL*8 AL1, AL2, AL3, AL4, AL5
REAL*8 BL1, BL2, BL3, BL4, BL5
REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
REAL*8 XW, YW, ZW
COMMON TIMAT,THETA

C Initialize the TIMAT matrix to an I matrix:

DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C open data file

OPEN (9, NAME='posever.DAT',STATUS='OLD')
OPEN (10, NAME='input.DAT', STATUS='OLD')
OPEN (11, NAME='result.DAT', STATUS='OLD')

c read input parameters

read (10,*)
read (10,*) world(1),world(2),world(3)
read (10,*) dt1,dd1,aa1,al1,bl1
read (10,*) dt2,dd2,aa2,al2,bl2
read (10,*) dt3,dd3,aa3,al3,bl3
read (10,*) dt4,dd4,aa4,al4,bl4
read (10,*) dt5,dd5,aa5,al5,bl5
read (10,*)
read (10,*) df6,th6,si6,px6,py6,pz6
read (10,*)
read (10,*) npos,r,dangle,dlenth,magnx,magnl

CLOSE (10)

c Read in joint angle sets for verification poses

```

```

      read (9,*) nposes

      do i=1,nposes
        read(9,*)theta(i,1),theta(i,2),theta(i,3),theta(i,4),
&          theta(i,5),theta(i,6)
      enddo
      close (9)

```

C Set exact link parameters for the manipulator:

```

      dt(1) = dt1          ! defined
      dt(2) = dt2 + dangle
      dt(3) = dt3          ! defined
      dt(4) = dt4 + dangle
      dt(5) = dt5 + dangle

      world(1) = world(1) + dlenth
      world(2) = world(2) + dlenth

      al(1) = al1 + DANGLE
      al(2) = al2 + DANGLE
      al(3) = al3 + DANGLE
      al(4) = al4 + DANGLE
      al(5) = al5 + DANGLE

      AA(1) = aa1 + DLENTH
      AA(2) = aa2 + DLENTH
      AA(3) = aa3          ! defined
      AA(4) = aa4 + DLENTH
      AA(5) = aa5 + DLENTH

      DD(1) = dd1 + DLENTH
      DD(2) = dd2 + DLENTH
      DD(3) = dd3 + DLENTH
      DD(4) = dd4          ! defined
      DD(5) = dd5 + DLENTH

      BL(1) = bl1          ! defined
      BL(2) = bl2          ! defined
      BL(3) = bl3 + DANGLE
      BL(4) = bl4          ! defined
      BL(5) = bl4          ! defined

      DF6 = DF6 + Dangle
      TH6 = 0.0
      SI6 = 0.0
      PX6 = PX6 + DLENTH
      PY6 = 0.0
      FZ6 = PZ6 + DLENTH

```

c Read in and set up estimated parameter table

```

      read(11,*)
      read(11,*)
      read(11,*) eworld(1),eworld(2),eworld(3)

      do i=1,5
        read (11,*)
        read (11,*)
        read (11,*) edt(i),edd(i),eaa(i),eal(i),ebl(i)
      enddo

```

```

      read(11,*)
      read(11,*)
      read(11,*) edf6,eth6,esi6,epx6,epy6,epz6,r

c    do kk=1,3
c    write(6,*)world(kk),eworld(kk)
c    enddo

c    do ii=1,6
c    write(6,*) ii
c    write(6,*)al(ii),eal(ii),aa(ii),eaa(ii),dd(ii),edd(ii),
c      & bl(ii),ebl(ii),dt(ii),edt(ii)
c    enddo

c Main loop through NPOSES joint angle sets

      do k=1,nposes

      call fks (k,world,dt,al,aa,dd,bl,fi6,th6,si6,px6,py6,pz6,t)
      call fks (k,eworld,edt,eal,eaa,edd,ebl,efi6,eth6,esi6,epx6,
      &          epy6,epz6,et)

c Compute the differential tool matrix

      call matsub (tdelta,t,et)

c Compute the pose errors

      poserr=sqrt(tdelta(1,4)**2+tdelta(2,4)**2+tdelta(3,4)**2)
      orerr1=(tdelta(3,2)-tdelta(2,3))/2
      orerr2=(tdelta(1,3)-tdelta(3,1))/2
      orerr3=(tdelta(2,1)-tdelta(1,2))/2
      orerr=sqrt(orerr1**2+orerr2**2+orerr3**2)

c Update total error counts

      posterr=(poserr+(k-1)*posterr)/k
      orterr =(orerr +(k-1)*orterr)/k

c End of main loop

      enddo

      write (6,*) 'Position error, orientation error'
      write (6,*) posterr,orterr

      OPEN (19, NAME='VER.DAT', STATUS='OLD')
      READ (19,*) NR
      IF(NR.GT.0) READ(19,*) (P(I),OR(I),W1(I),W2(I),W3(I),I=1,NR)
      NR=NR+1
      p(nr)=POSTERR
      or(nr)=ORTERR
      w1(nr)=WORLD(1)-DLENTN
      w2(nr)=WORLD(2)-DLENTN
      w3(nr)=WORLD(3)
      REWIND 19
      WRITE(19,*) nr
      WRITE(19,*) (P(I),OR(I),W1(I),W2(I),W3(I),I=1,NR)

```

```

        CLOSE (19)

    end

C *****

        subroutine fks (n,world,dt,al,aa,dd,bl,df6,th6,si6,
    &                    px6,py6,pz6,t)

        REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
        REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
        REAL*8 TIMAT(4,4), T(4,4), dt(5),al(5),aa(5),dd(5),bl(5)
        real*8 theta(1000,6), ang(5), world(3)
        common timat,theta

C Initialize the T matrix to an I matrix:

        DO J=1,4
        DO K=1,4
            T(J,K) = TIMAT(J,K)
        ENDDO
        ENDDO

C Set up the joint angles

        do i=1,5
            ang(i)=theta(n,i)
        enddo

        fi6=theta(n,6)+df6

C Compute the T matrices, T1 thru T6:

        call t3xyz (world(1),world(2),world(3),T0)

        CALL TRANSFORM (al(1),aa(1),dd(1),ang(1),bl(1),T1)
        CALL TRANSFORM (al(2),aa(2),dd(2),ang(2),bl(2),T2)
        CALL TRANSFORM (al(3),aa(3),ang(3),dt(3),bl(3),T3)
        CALL TRANSFORM (al(4),aa(4),dd(4),ang(4),bl(4),T4)
        CALL TRANSFORM (al(5),aa(5),dd(5),ang(5),bl(5),T5)

        CALL T3RPY (fi6,th6,si6,TRPY )
        CALL T3XYZ (px6,py6,pz6,TXYZ )
        CALL MATMULC ( T6, TRPY, TXYZ )

C Compute the overall transformation, T:

        CALL MATMULA ( T, T0 )
        CALL MATMULA ( T, T1 )
        CALL MATMULA ( T, T2 )
        CALL MATMULA ( T, T3 )
        CALL MATMULA ( T, T4 )
        CALL MATMULA ( T, T5 )
        CALL MATMULA ( T, T6 )

        return
    end

C *****

```

LIST OF REFERENCES

1. Pathre, U.S., and Driels, M.R., "Simulation Experiments in Parameter Identification for Robot Calibration," *The International Journal of Advanced Manufacturing Technology*, 1989.
2. Mooring, B., Roth, Z., and Driels, M., *Fundamentals of Robot Calibration*, John Wiley and Sons, Inc., 1989.
3. Jarvis, J., "Microsurveying: Towards Robot Accuracy," *Proc. 1987 IEEE International Conference on Robotics and Automation*, pp. 1660-1665, March-April 1987.
4. Lau, K., Hocken, R., and Haynes, L., "Robot Performance Measurements Using Automatic Laser Tracking Techniques," *Robotics and Computer Integrated Manufacturing*, Vol. 2, pp. 227-236, 1985.
5. Mooring, B., and Padavala, S., "The Effect of Model Complexity on Manipulator Accuracy," *Proc. 1989 IEEE International Conference on Robots and Automation*, pp. 593-598, May 1989.
6. Denavit, J., and Hartenberg, R., "A Kinematic Notation for Lower Pair Mechanism Based on Matrices," *ASME Journal Applied Mechanics*, Vol. 77, pp. 215-221, June 1955.
7. Hayati, S., "Robot Arm Geometric Link Parameter Estimation," *Proc. 22nd IEEE Conference on Decision and Control*, pp. 1477-1483, December 1983.
8. Mooring, B., "The Effect of Joint Axis Misalignment on Robot Positioning Accuracy," *Proc., ASME 1983 International Computers in Engineering Conference*, pp. 151-155, 1983.
9. Wu, C., "The Kinematic Error Model for the Design of Robot Manipulators," *Proc., American Control Conference*, 497-502, 1983.
10. Paul R., *Robot Manipulators: Mathematics, Programming, and Control*, pp. 45-47, MIT Press, 1982.

11. Everett, L., and Hsu, T., "The Theory of Kinematic Parameter Identification for Industrial Robots," *Trans ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 110, pp. 96-99, 1988.
12. Veitschegger, W.K., and Wu, C., "Robot Accuracy Analysis," *IEEE Journal of Robotics and Automation*, pp. 425-426, 1987.

INITIAL DISTRIBUTION LIST

	No. Copies	
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2	
2. Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2	✓
3. Department Chairman, Code ME Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940-5000	2	
4. Prof. M.R. Driels, Code ME/Dr Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940-5000	4	
5. LT M.J. Wiegand, USN Ship Repair Facility Subic Bay, RP Box 34 FPO San Francisco 96651-1400	2	
6. Naval Ocean Systems Center Code 531 P.O. Box 997 Kailua, HI 96734-0997	2	
7. Naval Engineering, Code 34 Naval Postgraduate School Monterey, CA 93943-5100	1	✓